

## REPORT DOCUMENTATION PAGE

AFRL-SR-BL-TR-01-

Public reporting burden for this collection of information is estimated to average 1 hour per response, including gathering and maintaining the data needed, and completing and reviewing the collection of information. Send collection of information, including suggestions for reducing this burden, to Washington Headquarters Service: Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project 0704-0188.

es,  
his  
son

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED	
	22 Aug 00	Final Tech Report 15 Nov 96 to 14 Nov 99	
4. TITLE AND SUBTITLE		5. FUNDING NUMBERS	
Development and Application of New Algorithms for the Simulation of Viscous Compressible Flows with Moving Bodies in Three Dimensions		F49620-97-1-0032	
6. AUTHOR(S)		8. PERFORMING ORGANIZATION REPORT NUMBER	
Rainald Lohner, Chi Yang and Juan R. Cebral			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)		9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	
George Mason University CSI Fairfax, VA 22030-4444		Air Force Office of Scientific Research AFOSR/NM 801 N. Randolph Street, Rm 732 Arlington, VA 22203-1977	
10. SPONSORING/MONITORING AGENCY REPORT NUMBER			
F49620-97-1-0032			
11. SUPPLEMENTARY NOTES			
12a. DISTRIBUTION AVAILABILITY STATEMENT		12b. DISTRIBUTION CODE	
Approved for public release; distribution unlimited.		AIR FORCE OFFICE OF SCIENTIFIC RESEARCH (AFOSR) NOTICE OF TRANSMITTAL DTIC. THIS TECHNICAL REPORT HAS BEEN REVIEWED AND IS APPROVED FOR PUBLIC RELEASE LAW AFR 180-12. DISTRIBUTION IS UNLIMITED.	
13. ABSTRACT (Maximum 200 words)			
The overall objective of the research carried out over the last three years was the development of new algorithms for the efficient simulation of viscous compressible flows with moving bodies in three dimensions using unstructured grids. The development was based on current 3-D Euler/Navier-Stokes capabilities, and encompassed flow solvers, grid generation, fluid-structure interaction, the efficient use of supercomputer hardware, and new visualization capabilities. The research carried out over the last three years significantly advanced the state of the art in this area of CFD. The particular topics are treated below in detail.			
14. SUBJECT TERMS		15. NUMBER OF PAGES	
16. PRICE CODE			
17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

FINAL REPORT: DECEMBER 1999

F 49620-97-1-0032

DEVELOPMENT AND APPLICATION OF NEW ALGORITHMS FOR  
THE SIMULATION OF VISCOUS COMPRESSIBLE FLOWS  
WITH MOVING BODIES IN THREE DIMENSIONS

Rainald Löhner, Chi Yang and Juan R. Cebral  
GMU/CSI, George Mason University  
Fairfax, VA 22030-4444

presented to:

Air Force Office of Scientific Research /NM  
c/o Dr. Leonidas Sakell  
801 North Randolph Street, Room 732  
Arlington VA 22203-1977

FINAL REPORT: DECEMBER 1999

F 49620-97-1-0032

DEVELOPMENT AND APPLICATION OF NEW ALGORITHMS FOR  
THE SIMULATION OF VISCOUS COMPRESSIBLE FLOWS  
WITH MOVING BODIES IN THREE DIMENSIONS

Rainald Löhner, Chi Yang and Juan R. Cebral  
GMU/CSI, George Mason University  
Fairfax, VA 22030-4444

SUMMARY

The overall objective of the research carried out over the last three years was the development of new algorithms for the efficient simulation of viscous compressible flows with moving bodies in three dimensions using unstructured grids. The development was based on current 3-D Euler/Navier-Stokes capabilities, and encompassed flow solvers, grid generation, fluid-structure interaction, the efficient use of supercomputer hardware, and new visualization capabilities. The research carried out over the last three years significantly advanced the state of the art in this area of CFD. The particular topics are treated below in detail.

## 1. FLOW SOLVERS

For the flow solvers, seven major developments took place over the course of this research effort:

- a) Implicit flow solvers;
- b) Better mesh moving strategies;
- c) Implementation of turbulence models; and
- d) Validation studies.

### 1.1 Implicit Flow Solvers

Implicit flow solvers are considered essential for the efficient simulation of viscous, compressible, time-dependent flows. We developed a linearized implicit scheme that uses a Generalized Minimal RESiduals algorithm in conjunction with incomplete lower-upper (ILU) preconditioning for the solution of the Euler and Navier-Stokes equations. The results were encouraging, showing that for Euler problems steady state results could be achieved in less than 40 steps. On the other hand, the storage costs and the cost of getting close to the solution at the start of the iteration were considered suboptimal. This led to development of matrix-free preconditioners, in particular GMRES-LU-SGS. The Euler/Navier-Stokes equations may be written as a system of conservation laws of the form:

$$\mathbf{u}_{,t} + \nabla \cdot \mathbf{F} = 0 \quad ,$$

or, in integral form, as:

$$\frac{d}{dt} \int \mathbf{u} d\Omega + \int \mathbf{F} \cdot \mathbf{n} d\Gamma = 0$$

The set of equations resulting from spatial discretization using an edge-based data structure may be written as:

$$\mathbf{M}_l^i \mathbf{u}_{,t}^i = \mathbf{r}^i = \sum \mathbf{C}^{ij} \cdot \mathbf{F}_{ij} \quad , \quad (1)$$

where  $\mathbf{M}$ ,  $\mathbf{C}$ ,  $\mathbf{u}$  and  $\mathbf{F}$  denote, respectively, the lumped mass-matrix (volume), geometry (shape-function derivative) factor, unknowns and fluxes. Work continued on the GMRES-LU-SGS scheme. This scheme is based on the vectorized LU-SGS scheme first proposed in [2]. The implicit time integration of Eqn.(1) using a backward Euler scheme with linearization yields:

$$\left[ \frac{1}{\Delta t} \mathbf{M}_l^i - \sum \mathbf{C}^{ij} \cdot \mathbf{A}_{ij} \right] \Delta \mathbf{u}^i = \mathbf{r}^i \quad . \quad (2)$$

Denoting by  $\rho_A$  the spectral radius of the Jacobian  $\mathbf{A}$  and defining:

$$\mathbf{D} = \left[ \frac{1}{\Delta t} \mathbf{M}_l^i - 0.5 \sum \mathbf{C}^{ij} \rho_A \right] \mathbf{I} \quad , \quad \Delta \mathbf{F} = \mathbf{F}(\mathbf{u} + \Delta \mathbf{u}) - \mathbf{F}(\mathbf{u}) \quad , \quad (3)$$

this system of equations is solved by the following relaxation procedure:

a) Forward Sweep:

$$\Delta \hat{\mathbf{u}}^i = \mathbf{D}^{-1} \left[ \mathbf{r}^i - 0.5 \sum \mathbf{C}^{ij} \cdot (\Delta \hat{\mathbf{F}}_{ij} - \rho_A \Delta \hat{\mathbf{u}}_j) \right] \quad (4a)$$

b) Backward Sweep:

$$\Delta \mathbf{u}^i = \Delta \hat{\mathbf{u}}^i - 0.5 \mathbf{D}^{-1} \sum \mathbf{C}^{ij} \cdot (\Delta \mathbf{F}_{ij} - \rho_A \Delta \mathbf{u}_j) \quad (4b)$$

We have shown in [2,5,6] that this scheme is an excellent preconditioner for the Generalized Minimal RESiduals (GMRES) iterative solver that is by now firmly established in CFD. This scheme was extended to transient problems (time-accurate solvers [6]) and was investigated further in two directions, namely: robustness and parallelism. The investigations with respect to the **robustness** of the GMRES-LU-SGS showed that the scheme was remarkably insensitive to point and edge numbering. **Parallel** schemes were investigated for shared-memory machines. It was found that the GMRES-LU-SGS parallelizes well up to 24 processors, the maximum number available to us at the present time. For technical details the reader is referred to [2], which is reproduced in Appendix 1.

## 1.2 Better Mesh Moving Strategies

A Laplacian smoothing of the mesh velocities with variable diffusivity based on the distance from moving bodies was developed [11]. This variable diffusivity enforces a more uniform mesh velocity in the region close to the moving bodies. Given that in most applications these are regions where small elements are located, the new procedure decreases considerably element distortion, reducing the need for local or global remeshing, and in some cases avoiding it altogether. A hypersonic store release was used to test the new algorithm. Numerical results obtained show that the new mesh velocity smoothing leads to a much less deformed grid close to the moving missile. For this case, the number of local remeshings required dropped by a factor of 1:4, leading to considerable CPU savings in multiprocessor environment. Since then, this algorithm has been used extensively for many applications [9-12]. For technical details the reader is referred to [11].

## 1.3 Implementation of Turbulence Models

We continued with the implementation of several turbulence models. Some of these have very stiff source or production terms that require careful numerical treatment. For technical details the reader is referred to [1,8].

## 1.4 Validation Studies

The efficiency and fidelity of the new Arbitrary Lagrangian-Eulerian (ALE) methodology on unstructured grids was validated by two simulations. This validation effort was part of an ongoing research effort to develop a cost-efficient and accurate numerical methodology capable of simulating the motion of complex-geometry, three-dimensional bodies embedded in external, temporally and spatially evolving flow-fields.

The first computation modeled the separation of a fuel tank from an F-16 C/D fighter. The numerical Eulerian predictions were compared to the available experimental data. First, a series of steady-state runs were performed to compare overall loads and moments. Then, the fuel tank was released, with proper modeling of the initial impulse due to explosive release. Very good agreement was obtained between the predicted and measured fuel tank trajectory [10].

The second simulation modeled canopy trajectory for an F-18 fighter. As before, very good agreement was obtained between the predicted and measured fuel tank trajectory [12].

## 2. GRID GENERATION

In the area of grid generation, there were three major developments that took place during the course of this research effort:

- a) Surface meshing from discrete data;
- b) Parallel grid generation; and
- c) Navier-Stokes gridding.

### 2.1 Surface Meshing from Discrete Data

An advancing front surface gridding technique that operates on discretely defined faces was developed [29]. This technique is based on three steps: surface feature recovery, actual gridding, and surface recovery. The following aspects have to be considered carefully in order to make the procedure reliable for complex geometries:

- a) Recovery of surface features and discrete surface patches from the discrete data,
- b) Filtering based on point and side normals to remove undesirable data close to cusps and corners,
- c) Proper choice of host faces for ridges, and
- d) Fast interpolation procedures suitable for complex geometry.

Several examples ranging from academic to industrial demonstrated the utility of the developed procedure for ab initio surface meshing from discrete data, such as encountered when the surface description is already given as discrete, the improvement of existing surface triangulations, as well as remeshing applications during runs exhibiting significant change of domain. For technical details the reader is referred to [29].

## 2.2 Parallel Grid Generation

FY99 saw the development, coding and debugging of a new parallel advancing front grid generation algorithm. We consider this to be truly a breakthrough, as it allows the unstructured grid ALE moving body methodology to be extended fully to parallel machines. To date, the required remeshing that always appears for moving mesh simulations with complex body motion had to be carried out in scalar mode. If we define as  $d_{min}$  the minimum element size in the active front, and as  $s_{min}$  the minimum box size in which elements are to be generated, the parallel advancing front grid generator proceeds as follows:

**WHILE:** There are active faces left:

- Form an octree with minimum octant size  $s_{min}$  for the active points;
- Retain the octants that have faces that will generate elements of size  $d_{min}$  to  $c_l \cdot d_{min}$ ;
- If too many octants are left: agglomerate them into boxes;
- **DO ISHFT=0,2:**
  - **IF:** ISHFT.NE.0:  
Shift the boxes by a preset amount;
  - **ENDIF**
  - Generate, in parallel, elements in these boxes, allowing only elements up to a size of  $c_l \cdot d_{min}$ ;
- **ENDDO**

- Increase  $d_{min} = 1.5 * d_{min}$ ,  $s_{min} = 1.5 * s_{min}$ ;

**ENDWHILE**

The increase factor allowed is typically in the range  $c_l = 1.5 - 2.0$ . Major areas of work included:

noi

- Treatment of cases with large variation of element size;
- Proper estimation of work per domain/processor;
- Load balancing; and
- Reduction of inter-box faces.

The parallel mesh generation technique was debugged and improved during FY99, and has now been incorporated into the production code FEFLO98. The speedups obtained for grid generation are now comparable to those of the CFD solver. For more details on the parallel grid generator, see [23], which is reproduced in Appendix 2.

### 2.3 Navier-Stokes Gridding

Creating highly stretched grids of acceptable quality for complex configurations has been an outstanding goal for over two decades. During FY98 and FY99 we continued the development of a new RANS gridding algorithm that was conceived in FY97. The key steps of this algorithm may be summarized as follows:

- a) Generate first an isotropic mesh.
- b) Using a constrained Delaunay technique, introduce points in order to generate highly stretched elements.
- c) Introduce the points in ascending level of stretching, i.e. from the domain interior to the boundary (or interior boundaries).

This procedure has the following advantages:

- No surface recovery is required for the Delaunay reconnection, eliminating the most problematic part of this technique;
- Proper meshing at concave ridges/corners is obtained;
- The meshing of concave ridges/corners requires no extra work;
- Meshing problems due to surface curvature are minimized;
- In principle, no CAD representation of the surface is required; and
- A final mesh is guaranteed, an essential requirement for industry.

The disadvantages are the following:

- As with any Delaunay technique, the mesh quality is extremely sensitive to point placement.

Major areas of work included:

- Point removal techniques in the pre-RANS-gridding phase;
- Proper point placement to mitigate the possible negative effects of Delaunay reconnection;
- Acceptance/rejection tests for new points, particularly for gaps;
- Reconnection of surface faces for optimal meshes; and
- Smooth transition of element size and stretching.

While this is work in progress, we feel confident that a new level of gridding technology has been achieved. For more details the reader is referred to [21,22], which is reproduced in Appendix 3.

### 3. FLUID-STRUCTURE-THERMAL COUPLING

In the area of interdisciplinary coupling, there were three major developments that took place during the course of this research effort:

- a) Loose coupling strategies for fluid-structure-thermal problems;
- b) Link to FEEIGEN, NASTRAN and DYNA3D; and
- c) Breakup and topology change.

#### 3.1 Loose Coupling Strategies

In order to solve, in a cost-effective manner, fluid-structure-thermal interaction problems, a loosely coupled algorithm to combine computational fluid dynamics (CFD), computational structural dynamics (CSD) and computational thermo-dynamic (CTD) codes was devised. In this algorithm, the structure is used as the 'master-surface' to define the extent of the fluid region, and the fluid is used as the 'master-surface' to define the loads. The transfer of loads, heat fluxes, displacements, velocities and temperatures is carried out via fast interpolation and projection algorithms. This fluid-structure-thermal algorithm can be interpreted as an iterative solution to the fully coupled, large matrix problem that results from the discretization of the complete problem. The advantage of this new algorithm is that it allows a cost effective re-use of existing software, with minimum amount of alterations required to account for the interaction of the different media.

Several example runs using FEFLO98 as the CFD code, and NASTRAN as the CSD/CTD code, demonstrate the effectiveness of the proposed methodology. For more details, see the AIAA invited paper, Ref. [15], which is reproduced in Appendix 4, as well as [13-19].

#### 3.2 Link to FEEIGEN, NASTRAN and DYNA3D

The CFD code was linked to three different structural solvers that span a large range of applications:

- a) FEEIGEN is an eigenmode integrator with adaptive timestep. FEEIGEN can read eigenmode information from a variety of sources, including such well-known codes as NASTRAN and ANSYS. It then integrates the eigenmodes in time, thus producing the structural response.
- b) NASTRAN is a general-purpose linear Finite Element code used in the present context for elasticity and thermal problems. The particular version used is the COSMIC-NASTRAN code, as it offered the possibility of accesing the source code.
- c) DYNA3D is a general-purpose nonlinear Finite Element code used in the present context for impact and shock-structure interaction simulation. The particular sed is the LLNL public-domain DYNA3D, as it offered the possibility of accesing the source code.

### 3.3 Breakup and Topology Change

The loosely coupled algorithm was extended to include breakup and topology change. Suppose that due to cracking, failure, spalation, etc., the ‘wetted surface’ of the CSD domain has been changed. This new surface, given by a list of points and faces, has to be matched with a corresponding CFD surface. The CFD surface data typically consists of surface segments defined by analytical functions that do not change in time (such as exterior walls, farfield boundaries, etc.), and surface segments defined by triangulations (i.e. discrete data) that change in time. These triangulations are obtained from the ‘wetted CSD surface’ at every timestep. When a change in topology is detected, the new surface definition is recovered from the discrete data, and joined to the surfaces defined analytically. The discrete surface is defined by a support triangulation, with lines and end-points to delimit its boundaries. In this sense, the only difference with analytically defined surfaces is the (discrete) support triangulation. The patches, lines and end-points of the ‘wetted CSD surface’ are identified by comparing the unit surface normals of adjacent faces. If the scalar product of them lies below a certain tolerance, a ridge is defined. **Corners** are defined as points that are attached to:

- Only one ridge;
- More than two ridges; or
- Two ridges with considerable deviation of unit side-vector.

Between corners, the ridges form **discrete lines**. These discrete lines either separate or are embedded completely (i.e. used twice) in **discrete surface patches** [19]. For the old surface definition data set, the surface patches attached to wetted CSD surfaces are identified and all information associated with them is discarded. The remaining data is then joined to the new wetted CSD surface data, producing the updated surface definition data set. This data set is then used to generate the new surface and volume grids.

The surface reconstruction procedure may be summarized as follows:

- For the Updated Discrete Data, Obtain:
  - Surface Patches + B.C.
  - Lines
  - End-Points
  - Sources
- For the Old Analytical+Discrete Data:
  - Remove Discrete Data
  - Reorder Arrays
- Merge:
  - Old Analytical Data
  - Updated Discrete Data

Once a new mesh has been generated, the solution from the previous timestep (on the previous mesh) has to be interpolated. Optimal interpolation algorithms for unstructured grids were developed in a previous AFOSR-sponsored effort. Whenever new

fluid domains are created due to failure, cracking and spalation, interpolating the fluid solution from the previous timestep to these new domains will end in failure, as there are no possible host elements in the old mesh. It is therefore important to identify points of the new mesh that lie outside the confines of the old mesh. A new way that was developed and that has proven successful is to form a Cartesian mesh or bins. A loop is then performed over the elements of the old mesh, marking the bins covered by elements. In a second loop over the points of the new grid, all points that fall into bins not covered by the old grid are marked as impossible to interpolate. This procedure can be done recursively by obtaining the confines of the volume where points have been marked as impossible, leading to so-called ‘telescoping’ of the bin search region. No attempt is then made to interpolate the points marked as outside the old mesh. The unknowns for these points can be extrapolated using different procedures:

- Advancing layers (most often used for subsonic/isotropic flows);
- Upstream (used primarily for supersonic flows);
- Closest known point (for cracks); or
- Via user-prescribed subroutine (for special cases).

For technical details the reader is referred to [19], which is reproduced in Appendix 5.

#### 4. EFFICIENT USE OF SUPERCOMPUTING HARDWARE

During the present effort we parallelized the majority of the ‘core’ subroutines required by the flow solver for shared memory, parallel machines. In order to appreciate the task at hand, consider the following Laplacian right-hand-side loop in its original form:

```

do 1600 iedge=1,nedge
  ipoi1=lnoed(1,iedge)
  ipoi2=lnoed(2,iedge)
  redge=geoed(  iedge)*(unkno(ipoi2)-unkno(ipoi1))
  rhspo(ipoi1)=rhspo(ipoi1)+redge
  rhspo(ipoi2)=rhspo(ipoi2)-redge
1600 continue

```

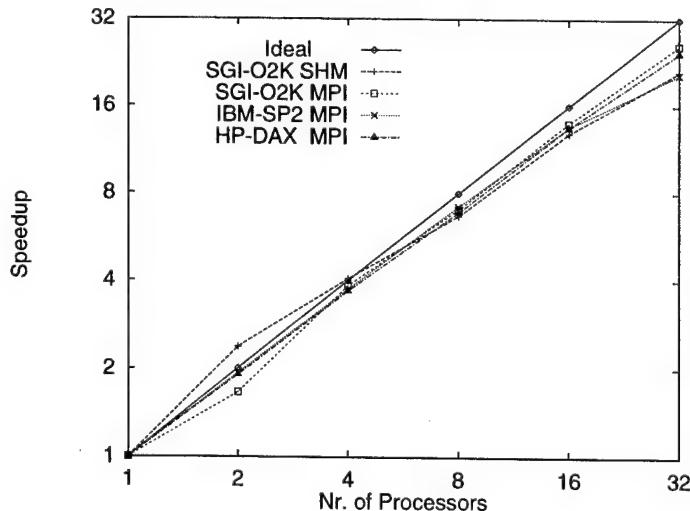
Here `nedge`, `lnoed`, `geoed`, `unkno`, `rhspo` denote, respectively, the number of edges, edge-point connectivity, geometrical and physical edge-parameters, unknowns at points and the right-hand-side vector at points. In order to run well on a shared memory, cache-and RISC-based (i.e. pipelined) parallel machine, it has to be re-written as:

```

do 1000 imacg=1,npasg,nproc
  imac0=           imacg
  imac1=min(npasg,imac0+nproc-1)
c$doacross local(ipasg,ipass,npas0,npas1,iedge,nedg0,nedg1,
c$&           ipoi1,ipoi2,redge)           ! Parallelization directive
  do 1200 ipasg=imac0,imac1
    npas0=edpag(ipasg)+1
    npas1=edpag(ipasg+1)
    do 1400 ipass=npas0,npas1
      nedg0=edpas(ipass)+1
      nedg1=edpas(ipass+1)
c$dir ivdep                                ! Pipelining directive
  do 1600 iedge=nedg0,nedg1
    ipoi1=lnoed(1,iedge)
    ipoi2=lnoed(2,iedge)
    redge=geoed( iedge)*(unkno(ipoi2)-unkno(ipoi1))
    rhspo(ipoi1)=rhspo(ipoi1)+redge
    rhspo(ipoi2)=rhspo(ipoi2)-redge
1600  continue
1400  continue
1200 continue
1000 continue

```

where `nproc` denotes the nr. of processors. Approximately 100 subroutines were re-written during this period of time. For technical details, see [24,25,26]. Figure 1 shows the speedups obtained for a supersonic inlet flow on a number of platforms. Scalability is evident.



Euler, 500Ktet, RK3, Roe+MUSCL

## 5. POST-PROCESSING

In order to handle the very large amounts of output data produced by current CFD runs with moving bodies, a new post-processing tool, called ZFEM, was developed. The following design requirements were set out for the new visualization system:

- Continuous (Grid-Based) and Discrete (Particle-Based) Data
- Distributed/Remote Reading
- Distributed/Remote Display
- Collaborative/Linked Display
- Multidisciplinary
- On-Line Display With Loose Coupling
- Interactive
- Repeatable (Movie-Making)
- Steering
- Portable
- User Friendly
- Minimum Storage
- Maximum Speed

A new visualization system was devised, whose core ideas are summarized below:

- Master/Worker/Viewer/Signaler Task Assignment
- Link Workers/Viewers on Different Hosts
- Interaction With Solver via Signal Files
- Script System (Movie-Making)
- Standard Libraries: OpenGL, Motif, PVM
- C Language
- Optimized Data Structures for Display
- GUI + Help System (HTML)

Two novel aspects, that set this new visualizer apart from anything hitherto developed, are:

- The parallel visualization of distributed data;
- The concurrent, collaborative visualization of data across the internet. This last aspect has made collaborative assignments across geographically distant locations a reality;
- The ability to connect, remotely, to a running flow solver for on-line visualization without disturbing its performance.

This new visualization system is still evolving. For more information on its current status, see the web-page: <http://www.science.gmu.edu/~jcebral/pages/zfem.html>.

## 6. SUMMARY

The present research effort significantly advanced the state of the art in the simulation of compressible viscous flows with moving bodies. A number of breakthroughs and 'firsts' were achieved, of which the following are considered the most important:

- The first fast matrix-free implicit scheme for unstructured grids (GMRES-LU-SGS) [2];
- The first scalable, shared-memory, unstructured-grid flow solver [25];
- The first conservative load transfer algorithm for fluid-structure interaction simulations [13];
- The first simulation of compressible flows with more than a thousand independently moving bodies [19];
- The most complex fluid-structure interaction simulation to date [16]; and
- The first fluid-structure interaction simulation capability for accurate fragmentation and cracking [19].

More work is still required to transform these algorithms into daily production tools that can be used effectively in the design and engineering process.

## 7. REFERENCES

We published extensively in the literature. Some of these papers are listed below.

### 7.1 Flow Solvers:

- [1] H. Luo, J.D. Baum and R. Löhner - Computation of Compressible Flows Using a Two-Equation Turbulence Model on Unstructured Grids; *AIAA-97-0430* (1997).
- [2] H. Luo, J.D. Baum and R. Löhner - A Fast, Matrix-Free Implicit Method for Compressible Flows on Unstructured Grids; *J. Comp. Phys.* 146, 664-690 (1998).
- [3] R. Löhner - Computational Aspects of Space-Marching; *AIAA-98-0617* (1998).
- [4] R. Löhner, J. Cebral, J.D. Baum and H. Luo - Capabilities and Issues of Unstructured-Grid CFD for High-Speed Flight Vehicles; *Int. CFD Workshop for Super-Sonic Transport Design*, Tokyo, Japan, March 16-17 (1998).
- [5] H. Luo, J.D. Baum and R. Löhner - A Fast, Matrix-Free Implicit Method for Compressible Flows on Unstructured Grids; *AIAA-99-0936* (1999).
- [6] H. Luo, J.D. Baum and R. Löhner - An Accurate, Fast, Matrix-Free Implicit Method for Computing Unsteady Flows on Unstructured Grids; *AIAA-99-0937* (1999).

- [7] H. Luo, J.D. Baum and R. Löhner - A Fast, Matrix-Free Implicit Method for Computing Low Mach-Number Flows on Unstructured Grids; *AIAA-99-3315-CP* (1999).
- [8] H. Luo, D. Sharov, J.D. Baum and R. Löhner - On the Computation of Compressible Turbulent Flows on Unstructured Grids; *AIAA-00-0926* (2000).
- [9] D. Sharov, H. Luo, J.D. Baum and R. Löhner - Implementation of Unstructured Grid GMRES+LU-SGS Method on Shared-Memory, Cache-Based Parallel Computers; *AIAA-00-0927* (2000).

## 7.2 Flow Solvers, Validation and Studies:

- [10] J.D. Baum, H. Luo, R. Löhner, E. Goldberg and A. Feldhun - Application of Unstructured Adaptive Moving Body Methodology to the Simulation of Fuel Tank Separation From an F-16 C/D Fighter; *AIAA-97-0166* (1997).
- [11] Chi Yang and R. Löhner - Numerical Simulation of a Maneuvering Missile Using a Loose Fluid-Structure Coupling Algorithm; *AIAA-97-0408* (1997).
- [12] J.D. Baum, R. Löhner, T.J. Marquette and H. Luo - Numerical Simulation of Aircraft Canopy Trajectory; *AIAA-97-1885* (1997).

## 7.3 Link to CSD:

- [13] J.R. Cebral and R. Löhner - Conservative Load Projection and Tracking for Fluid-Structure Problems; *AIAA J.* 35, 4, 687-692 (1997).
- [14] J.R. Cebral and R. Löhner - Fluid-Structure Coupling: Extensions and Improvements; *AIAA-97-0858* (1997).
- [15] R. Löhner, C. Yang, J. Cebral, J.D. Baum, H. Luo, D. Pelessone and C. Charman - Fluid-Structure-Thermal Interaction Using a Loose Coupling Algorithm and Adaptive Unstructured Grids; *AIAA-98-2419* [Invited] (1998).
- [16] J.D. Baum, H. Luo, E. Mestreau, R. Löhner, D. Pelessone and C. Charman - A Coupled CFD/CSD Methodology for Modeling Weapon Detonation and Fragmentation; *AIAA-99-0794* (1999).
- [17] R. Löhner, C. Yang, J. Cebral, J.D. Baum, E. Mestreau, H. Luo, D. Pelessone and C. Charman - Fluid-Structure-Thermal Interaction Using Adaptive Unstructured Grids; pp.109-120 in *Computational Methods for Fluid-Structure Interaction* (T. Kvamsdal et al. eds.), Tapir Press (1999).
- [18] R. Löhner, Chi Yang, J.D. Baum, H. Luo, D. Pelessone and C. Charman - The Numerical Simulation of Strongly Unsteady Flows With Hundreds of Moving Bodies; *Int. J. Num. Meth. Fluids* 31, 113-120 (1999).

[19] R. Löhner, C. Yang, J. Cebral, J.D. Baum, H. Luo, E. Mestreau, D. Pelessone and C. Charman - Fluid-Structure Interaction Algorithms for Rupture and Topology Change; *Proc. 1999 JSME Computational Mechanics Division Meeting*, Matsuyama, Japan, November (1999).

#### 7.4 Grid Generation:

[29] R. Löhner - Automatic Unstructured Grid Generators; *Finite Elements in Analysis and Design* 25, 111-134 (1997).

[21] R. Löhner - Automatic Generation of Unstructured Grids Suitable for RANS Calculations; *Proc. ICASE/LaRC/ARO/NSF Workshop on Computational Aero-sciences in the 21st Century*, Hampton, VA, April 22-24 (1998).

[22] R. Löhner - Generation of Unstructured Grids Suitable for RANS Calculations; *AIAA-99-0662* (1999).

[23] R. Löhner - A Parallel Advancing Front Grid Generation Scheme; *AIAA-00-1005* (2000).

#### 7.5 Supercomputing

[24] J. Tuszyński and R. Löhner - Parallelizing the Construction of Indirect Access Arrays for Shared-Memory Machines; *Comm. Appl. Num. Meth. Eng.* 14, 773-781 (1998).

[25] R. Löhner - Renumbering Strategies for Unstructured-Grid Solvers Operating on Shared-Memory, Cache-Based Parallel Machines; *Comp. Meth. Appl. Mech. Eng.* 163, 95-109 (1998).

[26] J.R. Cebral - ZFEM: Collaborative Visualization for Parallel Multidisciplinary Applications; *Proc. Parallel CFD'97*, Manchester, UK, May (1997).

#### 7.6 Visualization

[27] J.R. Cebral and R. Löhner - Interactive On-Line Visualization and Collaboration for Parallel Unstructured Multidisciplinary Applications; *AIAA-98-0077* (1998).

[28] J. Cebral and R. Löhner - Advances in Visualization: Distribution and Collaboration; *AIAA-99-0693* (1999).

## APPENDIX 1: IMPLICIT FLOW SOLVERS

JOURNAL OF COMPUTATIONAL PHYSICS 146, 664-690 (1998)  
ARTICLE NO. CP986076

## A Fast, Matrix-free Implicit Method for Compressible Flows on Unstructured Grids

Hong Luo, Joseph D. Baum, and Rainald Löhner

Reprinted from JOURNAL OF COMPUTATIONAL PHYSICS, Vol. 146, No. 2, November, 1998  
Copyright © 1998 by Academic Press, Inc. *Printed in U.S.A.*

# A Fast, Matrix-free Implicit Method for Compressible Flows on Unstructured Grids

Hong Luo,<sup>\*,†</sup> Joseph D. Baum,<sup>\*</sup> and Rainald Löhner<sup>†</sup>

<sup>\*</sup>*Applied Physics Operations, Science Applications International Corporation, McLean, Virginia 22102; †Institute for Computational Sciences and Informatics, George Mason University, Fairfax, Virginia 22030*  
E-mail: luo@mclapo.saic.com

Received October 21, 1997; revised July 31, 1998

A fast, matrix-free implicit method has been developed to solve the three-dimensional compressible Euler and Navier–Stokes equations on unstructured meshes. An approximate system of linear equations arising from the Newton linearization is solved by the GMRES (generalized minimum residual) algorithm with a LU-SGS (lower–upper symmetric Gauss–Seidel) preconditioner. A remarkable feature of the present GMRES+LU-SGS method is that the storage of the Jacobian matrix can be completely eliminated by approximating the Jacobian with numerical fluxes, resulting in a matrix-free implicit method. The method developed has been used to compute the compressible flows around 3D complex aerodynamic configurations for a wide range of flow conditions, from subsonic to supersonic. The numerical results obtained indicate that the use of the GMRES+LU-SGS method leads to a significant increase in performance over the best current implicit methods, GMRES+ILU and LU-SGS, while maintaining memory requirements similar to its explicit counterpart. An overall speedup factor from eight to more than one order of magnitude for all test cases in comparison with the explicit method is demonstrated. © 1998 Academic Press

## 1. INTRODUCTION

The use of unstructured meshes for computational fluid dynamics problems has become widespread due to their ability to discretize arbitrarily complex geometries and due to the ease of adaptation in enhancing the solution accuracy and efficiency through the use of adaptive refinement techniques. In recent years, significant progress has been made in

<sup>†</sup> Corresponding author.

developing numerical algorithms for the solution of the compressible Euler and Navier-Stokes equations on unstructured grids.

Early efforts in the development of temporal discretization methods using unstructured grids focused on explicit schemes. Usually, explicit temporal discretizations such as multistage Runge-Kutta schemes are used to drive the solution to steady state. Acceleration techniques such as local time-stepping and implicit residual smoothing have also been combined in this context. In general, explicit schemes and their boundary conditions are easy to implement, vectorize and parallelize, and require only limited memory storage. However, for large-scale problems and especially for the solution of the Navier-Stokes equations, the rate of convergence slows down dramatically, resulting in inefficient solution techniques. In order to speed up convergence, a multigrid strategy or an implicit temporal discretization is required.

In general, implicit methods require the solution of a linear system of equations arising from the linearization of a fully implicit scheme at each time step or iteration. The most widely used methods to solve a linear system on unstructured grids are iterative solution methods and approximate factorization methods. Significant efforts have been made to develop efficient iterative solution methods. These range from Gauss-Seidel to Krylov subspace methods that use a wide variety of preconditioners (see, e.g., Stoufflet [1], Batina [2], Venkatakrishnan *et al.* [3], Knight [4], Whitaker [5], Luo *et al.* [6], and Barth *et al.* [7]). The most successful and effective iterative method is to use the Krylov subspace methods [8, 9] such as GMRES and BICGSTAB with an ILU (incomplete lower-upper) factorization preconditioner. The drawback is that they require a considerable amount of memory to store the Jacobian matrix, which may be prohibitive for large problems. Recently, the lower-upper symmetric Gauss-Seidel method developed first by Jameson and Yoon [10] on structured grids has been successfully generalized and extended to unstructured meshes by several authors [11-13]. The most attractive feature of this approximate factorization method is that the evaluation and storage of the Jacobian matrix inherent in the original formulation of the LU-SGS method can be completely eliminated by making some approximations to the implicit operator. The resulting LU-SGS method can be made even cheaper than the explicit method per time step. However, this method is less effective than the most efficient iterative methods such as GMRES+ILU, because of slow convergence, requiring thousands of time steps to achieve a steady state.

The objective of the effort discussed in this paper is to develop a fast implicit method for solving compressible flow problems around 3D complex, realistic aerodynamic configurations on unstructured grids. Typically, hundreds of thousands of mesh points are necessary to represent such engineering-type configurations accurately. Any implicit methods requiring the storage of the Jacobian matrix would be impractical, if not impossible to use to solve such large-scale problems, where the storage requirement can easily exceed the memory limitation of present computers. In the present work a system of linear equations, arising from an approximate linearization of a fully implicit temporal discretization at each time step, is solved iteratively by a GMRES algorithm with an LU-SGS preconditioner. The idea behind this is to combine the efficiency of the iterative methods and low memory requirement of approximate factorization methods in an effort to develop a fast, low storage implicit method. An apparent advantage of the LU-SGS preconditioner is that it uses the Jacobian matrix of the linearized scheme as a preconditioner matrix, as compared with ILU preconditioner and, consequently, does not require any additional memory storage and computational effort to store and compute the preconditioner matrix. Furthermore,

the storage of the approximate Jacobian matrix can be completely eliminated by approximating the Jacobian with numerical fluxes, which will lead to a fast, low-storage implicit algorithm. The matrix-free implicit method developed has been used to compute a wide range of test problems and has been compared with a well-known GMRES+ILU algorithm and an approximately factored implicit algorithm LU-SGS. The new algorithm is found to offer substantial CPU time savings over the best current implicit methods, while maintaining a memory requirement competitive with the explicit method.

## 2. GOVERNING EQUATIONS

The Navier–Stokes equations governing unsteady compressible viscous flows can be expressed in the conservative form as

$$\frac{\partial \mathbf{U}}{\partial t} + \frac{\partial \mathbf{F}^j}{\partial \mathbf{x}_j} = \frac{\partial \mathbf{G}^j}{\partial \mathbf{x}_j}, \quad (2.1)$$

where the summation convention has been employed. The unknown vector  $\mathbf{U}$ , inviscid flux vector  $\mathbf{F}$ , and viscous flux vector  $\mathbf{G}$  are defined by

$$\mathbf{U} = \begin{pmatrix} \rho \\ \rho u_i \\ \rho e \end{pmatrix}, \quad \mathbf{F}^j = \begin{pmatrix} \rho u_j \\ \rho u_i u_j + p \delta_{ij} \\ u_j (\rho e + p) \end{pmatrix}, \quad \mathbf{G}^j = \begin{pmatrix} 0 \\ \sigma_{ij} \\ u_l \sigma_{lj} + k \frac{\partial T}{\partial x_j} \end{pmatrix}. \quad (2.2)$$

Here  $\rho$ ,  $p$ ,  $e$ ,  $T$ , and  $k$  denote the density, pressure, specific total energy, temperature, and thermal conductivity of the fluid, respectively, and  $u_i$  is the velocity of the flow in the coordinate direction  $x_i$ . This set of equations is completed by the addition of the equation of state,

$$p = (\gamma - 1)\rho \left( e - \frac{1}{2}u_j u_j \right), \quad T = \left( e - \frac{1}{2}u_j u_j \right) / C_v, \quad (2.3)$$

which is valid for perfect gas, where  $\gamma$  is the ratio of the specific heats, and  $C_v$  is the specific heat at constant volume. The components of the viscous stress tensor  $\sigma_{ij}$  are given by

$$\sigma_{ij} = \mu \left( \frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right) + \lambda \frac{\partial u_k}{\partial x_k} \delta_{ij}. \quad (2.4)$$

The thermal conductivity  $k$  and viscosity coefficient  $\mu$  are assumed to be a function of the temperature and are determined using Sutherland's empirical relation. It is assumed that  $\lambda$  and  $\mu$  are related by Stokes' hypothesis

$$\lambda = -\frac{2\mu}{3}. \quad (2.5)$$

The left-hand side of Eq. (2.1) constitutes the Euler equations governing unsteady compressible inviscid flows.

In the sequel, we assume that  $\Omega$  is the flow domain,  $\Gamma$  is its boundary, and  $\mathbf{n}_j$  is the unit outward normal vector to the boundary.

### 3. HYBRID DISCRETE FORMULATION

Assuming  $\Omega_h$  is a classical triangulation of  $\Omega$ ,  $N_I$  is a standard linear finite element shape function associated with a node  $I$ , and  $C_I$  is a dual mesh cell associated with the node, the hybrid finite volume and finite element formulation used for discretization of the Navier–Stokes equations is

find  $\mathbf{U}_h \in \mathcal{T}_h$  such that for each  $N_I$  ( $1 \leq I \leq n$ )

$$\int_{C_I} \frac{\partial \mathbf{U}_h}{\partial t} d\Omega + \int_{\partial C_I} \mathbf{F}^j(\mathbf{U}_h) \cdot \mathbf{n}_j d\Gamma = \int_{\Omega_h} \frac{\partial \mathbf{G}^j(\mathbf{U}_h)}{\partial \mathbf{x}_j} N_I d\Omega, \quad (3.1)$$

where  $\mathcal{T}_h$  is a discrete approximation space of suitable continuous functions. Inviscid fluxes are discretized using a cell-vertex finite volume formulation, where the control volumes are nonoverlapping dual cells constructed by the median planes of the tetrahedra. In the present study the numerical flux functions for inviscid fluxes at the dual mesh cell interface are computed using the AUSM+ [14] (advection upwind splitting method) scheme. A MUSCL [15] approach is used to achieve high-order accuracy. The Van Albada limiter based on primitive variables is used to suppress the spurious oscillation in the vicinity of the discontinuities. The implementation of the precise MUSCL strategy used in the present work can be found in Ref. [21]. Viscous flux terms are evaluated using a linear finite element approximation, which is equivalent to a second-order accurate central difference.

Equation (3.1) can then be rewritten in a semi-discrete form as

$$V_i \frac{\partial \mathbf{U}_i}{\partial t} = \mathbf{R}_i, \quad (3.2)$$

where  $V_i$  is the volume of the dual mesh cell (equivalent to the lumped mass matrix in the finite element), and  $\mathbf{R}_i$  is the right-hand side residual and equals zero for a steady-state solution.

### 4. IMPLICIT TIME INTEGRATION

In order to obtain a steady-state solution, the spatially discretized Navier–Stokes equations must be integrated in time. Using Euler implicit time-integration, Eq. (3.2) can be written in discrete form as

$$V_i \frac{\Delta \mathbf{U}_i^n}{\Delta t} = \mathbf{R}_i^{n+1}, \quad (4.1)$$

where  $\Delta t$  is the time increment and  $\Delta \mathbf{U}^n$  is the difference of an unknown vector between time levels  $n$  and  $n + 1$ ; i.e.,

$$\Delta \mathbf{U}^n = \mathbf{U}^{n+1} - \mathbf{U}^n. \quad (4.2)$$

Equation (4.1) can be linearized in time as

$$V_i \frac{\Delta \mathbf{U}_i^n}{\Delta t} = \mathbf{R}_i^n + \frac{\partial \mathbf{R}_i^n}{\partial \mathbf{U}} \Delta \mathbf{U}_i, \quad (4.3)$$

where  $\mathbf{R}_i$  is the right-hand side residual and equals zero for a steady-state solution. Writing the equation for all nodes leads to the delta form of the backward Euler scheme

$$A \Delta \mathbf{U} = \mathbf{R}, \quad (4.4)$$

where

$$A = \frac{V}{\Delta t} \mathbf{I} - \frac{\partial \mathbf{R}^n}{\partial \mathbf{U}}. \quad (4.5)$$

Note that as  $\Delta t$  tends to infinity, the scheme reduces to the standard Newton's method for solving a system of nonlinear equations. Newton's method is known to have a quadratic convergence property. The term  $\partial \mathbf{R} / \partial \mathbf{U}$  represents symbolically the Jacobian matrix. It involves the linearization of both inviscid and viscous flux vectors. In order to obtain the quadratic convergence of Newton's method, the linearization of the numerical flux function must be virtually exact. Unfortunately, explicit formation of the Jacobian matrix resulting from the exact linearization of any second-order numerical flux functions for inviscid fluxes can require excessive storage and is extremely expensive, if not impossible to evaluate. In order to reduce the number of nonzero entries in the matrix and to simplify the linearization, only a first-order representation of the numerical fluxes is linearized. This results in the graph of the sparse matrix  $\partial \mathbf{R} / \partial \mathbf{U}$  being identical to the graph of the supporting unstructured mesh. In addition, the following simplified flux function is used to obtain the left-hand side Jacobian matrix,

$$\mathbf{R}_i(\mathbf{U}_i, \mathbf{U}_j, \mathbf{n}_{ij}) = \frac{1}{2}(\mathbf{F}(\mathbf{U}_i, \mathbf{n}_{ij}) + \mathbf{F}(\mathbf{U}_j, \mathbf{n}_{ij})) - \frac{1}{2}|\lambda_{ij}|(\mathbf{U}_j - \mathbf{U}_i), \quad (4.6)$$

where

$$|\lambda_{ij}| = |\mathbf{V}_{ij} \cdot \mathbf{n}_{ij}| + C_{ij} + \frac{\mu_{ij}}{\rho_{ij}|x_j - x_i|}, \quad (4.7)$$

where  $\mathbf{n}_{ij}$  is the unit vector normal to the cell interface,  $\mathbf{V}_{ij}$  is the velocity vector, and  $C_{ij}$  is the speed of sound. Note that this flux function is derived by replacing the Roe's matrix by its spectral radius in the well-known Roe's Flux function [16],

$$\mathbf{R}_i^{\text{inv}} = \frac{1}{2}(\mathbf{F}(\mathbf{U}_i, \mathbf{n}_{ij}) + \mathbf{F}(\mathbf{U}_j, \mathbf{n}_{ij})) - \frac{1}{2}|J(\tilde{U})|(\mathbf{U}_j - \mathbf{U}_i) \quad (4.8)$$

for the inviscid flux vector, and the viscous Jacobian matrix is simply approximated by its spectral radius in the above linearization process. The linearization of flux function (4.6) yields

$$\frac{\partial \mathbf{R}_i}{\partial \mathbf{U}_i} = \frac{1}{2}(J(\mathbf{U}_i) + |\lambda_{ij}| \mathbf{I}) \quad (4.9)$$

$$\frac{\partial \mathbf{R}_i}{\partial \mathbf{U}_j} = \frac{1}{2}(J(\mathbf{U}_j) - |\lambda_{ij}| \mathbf{I}), \quad (4.10)$$

where  $J = \partial \mathbf{F} / \partial \mathbf{U}$  represents the Jacobian of the inviscid flux vector. The penalty for making these approximations in the linearization process is that the quadratic convergence of

Newton's method can no longer be achieved because of the mismatch and inconsistency between the right- and left-hand sides in Eq. (4.4). Although the number of time steps (Newton iterations, if  $\Delta t$  tends to infinity) may increase, the cost per each time step is significantly reduced: it takes less CPU time to compute the Jacobian matrix and the conditioning of the simplified Jacobian matrix is improved, thus reducing computational cost to solve the resulting linear system.

As only a first-order representation of the numerical fluxes is considered, the number of nonzero entries in each row of the matrix is related to the number of edges incident to the node associated with that row. In other words, each edge  $ij$  will guarantee nonzero entries in the  $i$ th column and  $j$ th row and, similarly, the  $j$ th column and  $i$ th row. In addition, nonzero entries will be placed on the diagonal of the matrix. Using an edge-based data structure, the left-hand side Jacobian matrix is stored in upper, lower, and diagonal forms, which can be expressed as

$$U = \frac{1}{2}(J(\mathbf{U}_j, \mathbf{n}_{ij}) - |\lambda_{ij}| \mathbf{I}), \quad (4.11)$$

$$L = \frac{1}{2}(-J(\mathbf{U}_i, \mathbf{n}_{ij}) - |\lambda_{ij}| \mathbf{I}), \quad (4.12)$$

$$D = \frac{V}{\Delta t} \mathbf{I} + \sum_j \frac{1}{2}(J(\mathbf{U}_i, \mathbf{n}_{ij}) + |\lambda_{ij}| \mathbf{I}). \quad (4.13)$$

Note that  $U$ ,  $L$ , and  $D$  represent the strict upper matrix, the strict lower matrix, and the diagonal matrix, respectively. Both upper and lower matrices require a storage of  $nedge \times neqns \times neqns$  and the diagonal matrix needs a storage of  $npoin \times neqns \times neqns$ , where  $npoin$  is the number of grid points;  $neqns$  (=5 in 3D) is the number of unknown variables and  $nedge$  is the number of edges. Note that in 3D  $nedge \approx 7npoin$ . Clearly, the upper and lower matrix consume substantial amounts of memory, taking 93% of the storage required for left-hand side Jacobian matrix.

Equation (4.4) represents a system of linear simultaneous algebraic equations and needs to be solved at each time step. The most widely used methods to solve this linear system are iterative solution methods and approximate factorization methods. Recently, the lower-upper symmetric Gauss–Seidel method developed first by Jameson and Yoon [10] on structured grids has been successfully generalized and extended to unstructured meshes by several authors [11–13]. The LU-SGS method is attractive because of its good stability properties and competitive computational cost in comparison to explicit methods. In this method, the matrix  $A$  is split in three matrices, a strict lower matrix  $L$ , a diagonal matrix  $D$ , and a strict upper matrix  $U$ . This system is approximately factored by neglecting the last term on the right-hand side of Eq. (4.14). The resulting equation can be solved in the two steps shown in Eqs. (4.15) and (4.16), each of them involving only simple block matrix inversions:

$$(D + L)D^{-1}(D + U)\Delta\mathbf{U} = \mathbf{R} + (LD^{-1}U)\Delta\mathbf{U}. \quad (4.14)$$

Lower (forward) sweep:

$$(D + L)\Delta\mathbf{U}^* = \mathbf{R}. \quad (4.15)$$

Upper (backward) sweep:

$$(D + U)\Delta\mathbf{U} = D\Delta\mathbf{U}^*. \quad (4.16)$$

Both lower and upper sweeps can be vectorized by appropriately reordering the grid points [13], resulting in a very efficient algorithm. It is found that the CPU cost of one LU-SGS step is approximately 50% of one three-stage Runge–Kutta explicit step.

It is clear that the above algorithm involves primarily the Jacobian matrix-solution incremental vector product. Such operation can be approximately replaced by computing increments of the flux vector  $\Delta\mathbf{F}$ :

$$J\Delta\mathbf{U} \approx \Delta\mathbf{F} = \mathbf{F}(\mathbf{U} + \Delta\mathbf{U}) - \mathbf{F}(\mathbf{U}). \quad (4.17)$$

This idea of the matrix-free approach, in which the product of Jacobian matrix and incremental vector is approximated by the increment of the flux vector, was first introduced in the work of Sharov and Nakahashi [13]. The forward sweep and backward sweep steps can then be expressed as

$$\Delta\mathbf{U}_i^* = D^{-1} \left[ \mathbf{R}_i - \sum_{j:j < i} \frac{1}{2} (\Delta\mathbf{F}_j^* - |\lambda_{ij}| \Delta\mathbf{U}_j^*) s_{ij} \right], \quad (4.18)$$

$$\Delta\mathbf{U}_i = \Delta\mathbf{U}_i^* - D^{-1} \sum_{j:j > i} \frac{1}{2} (\Delta\mathbf{F}_j - |\lambda_{ij}| \Delta\mathbf{U}_j) s_{ij}. \quad (4.19)$$

The most remarkable achievement of this approximation is that there is no need to store the upper and lower matrices  $U$  and  $L$ , which substantially reduces the memory requirements. It is found that this approximation does not compromise any numerical accuracy, and the extra computational cost is negligible.

Although the LU-SGS method is more efficient than its explicit counterpart, a significant number of time steps are still required to achieve the steady-state solution, due to the nature of the approximation factorization schemes. One way to speed up the convergence is to use iterative methods. In this work, the system of linear equations is solved by the generalized minimal residual (GMRES) method of Saad and Schultz [8]. This is a generalization of the conjugate gradient method for solving a linear system where the coefficient matrix is not symmetric and/or positive definite. The use of GMRES combined with different preconditioning techniques is becoming widespread in the CFD community for the solution of the Euler and Navier–Stokes equations [3, 5–7]. GMRES minimizes the norm of the computed residual vector over the subspace spanned by a certain number of orthogonal search directions. It is well known that the speed of convergence of an iterative algorithm for a linear system depends on the condition number of the matrix  $A$ . GMRES works best when the eigenvalues of matrix  $A$  are clustered. The easiest and the most common way to improve the efficiency and robustness of GMRES is to use preconditioning to attempt to cluster the eigenvalues at a single value. The preconditioning technique involves solving an equivalent preconditioned linear system,

$$\tilde{A}\Delta\tilde{\mathbf{U}} = \tilde{\mathbf{R}}, \quad (4.20)$$

instead of the original system (4.4), in the hope that  $\tilde{A}$  is well conditioned. Left preconditioning involves premultiplying the linear system with a matrix as

$$P^{-1}A\Delta\mathbf{U} = P^{-1}\mathbf{R}, \quad (4.21)$$

where  $P$  is the preconditioning matrix. The best preconditioning matrix for  $A$  would cluster as many eigenvalues as possible at unity. Obviously, the optimal choice of  $P$  is  $A$ , in which case the underlying matrix problem for GMRES is trivially solved with one Krylov vector. The motivation for preconditioning is twofold: (a) reduce the computational effort required to solve the linearized system of equations at each time-step; (b) reduce the total number of time steps required to obtain a steady state solution. Preconditioning will be cost-effective only if the additional computational work incurred for each subiteration is compensated for by a reduction in the total number of iterations to convergence. In this way, the total cost of solving the overall nonlinear system is reduced. In the present work, the LU-SGS presented above is used as a preconditioner, i.e.,

$$P = (D + L)D^{-1}(D + U). \quad (4.22)$$

A clear advantage of the LU-SGS preconditioner is that it uses the Jacobian matrix of the linearized scheme as a preconditioner matrix, as compared with ILU preconditioner. Consequently it does not require any additional memory storage and computational effort to store and compute the preconditioner matrix. The preconditioned restarted GMRES algorithm is described below.

ALGORITHM. Restarted preconditioned GMRES( $k$ ).

```

For  $l = 1, m$  do       $m$  restart iterations
   $\mathbf{v}_0 = \mathbf{R} - A\Delta\mathbf{U}_0$       initial residual
   $\mathbf{r}_0 := P^{-1}\mathbf{v}_0$       preconditioning step
   $\beta := \|\mathbf{r}_0\|_2$       initial residual norm
   $\mathbf{v}_1 := \mathbf{r}_0/\beta$       define initial Krylov
  for  $j = 1, k$  do      inner iterations
     $\mathbf{y}_j := A\mathbf{v}_j$       matrix–vector product
     $\mathbf{w}_j := P^{-1}\mathbf{y}_j$       preconditioning step
    For  $i = 1, j$  do      Gram–Schmidt step
       $h_{i,j} := (\mathbf{w}_j, \mathbf{v}_i)$       .
       $\mathbf{w}_j = \mathbf{w}_j - h_{i,j}\mathbf{v}_i$       .
    EndDo
     $h_{j+1,j} := \|\mathbf{w}_j\|_2$       .
     $\mathbf{v}_{j+1} := \mathbf{w}_j / h_{j+1,j}$       define Krylov vector
  EndDo
   $\mathbf{z} := \min_{\hat{\mathbf{z}}} \|\beta\mathbf{e}_1 - H\hat{\mathbf{z}}\|_2$       least squares solve
   $\Delta\mathbf{U} := \Delta\mathbf{U}_0 + \sum_{i=1}^m \mathbf{v}_i \mathbf{z}_i$       approximate solution
  if  $\|\beta\mathbf{e}_1 - H\mathbf{z}\|_2 \leq \epsilon$  exit      convergence check
   $\Delta\mathbf{U}_0 := \Delta\mathbf{U}$       restart
EndDo

```

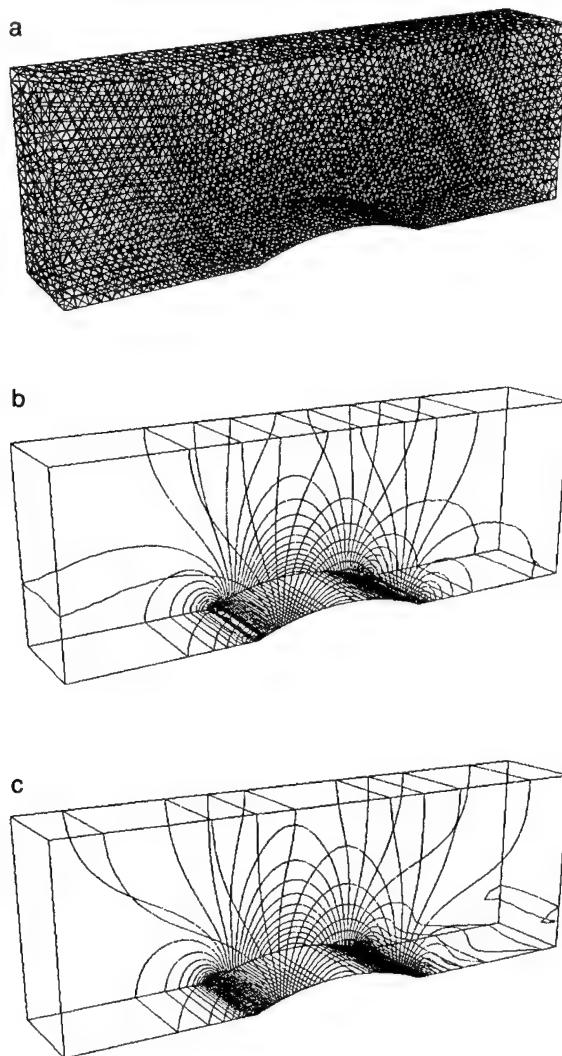
Note that the above GMRES algorithm only requires matrix–vector products, the same technique used in the LU-SGS method can be applied to eliminate the storage of the upper and lower matrices.

The present GMRES+LU-SGS method only requires the storage of the diagonal matrix. In addition, a storage corresponding to  $2 \times nedge$  is required for the two index arrays, which are necessary to achieve the vectorization of LU-SGS method. The need for additional storage associated with the GMRES algorithm is an array of size  $(k + 2) \times neqns \times npoin$ ,

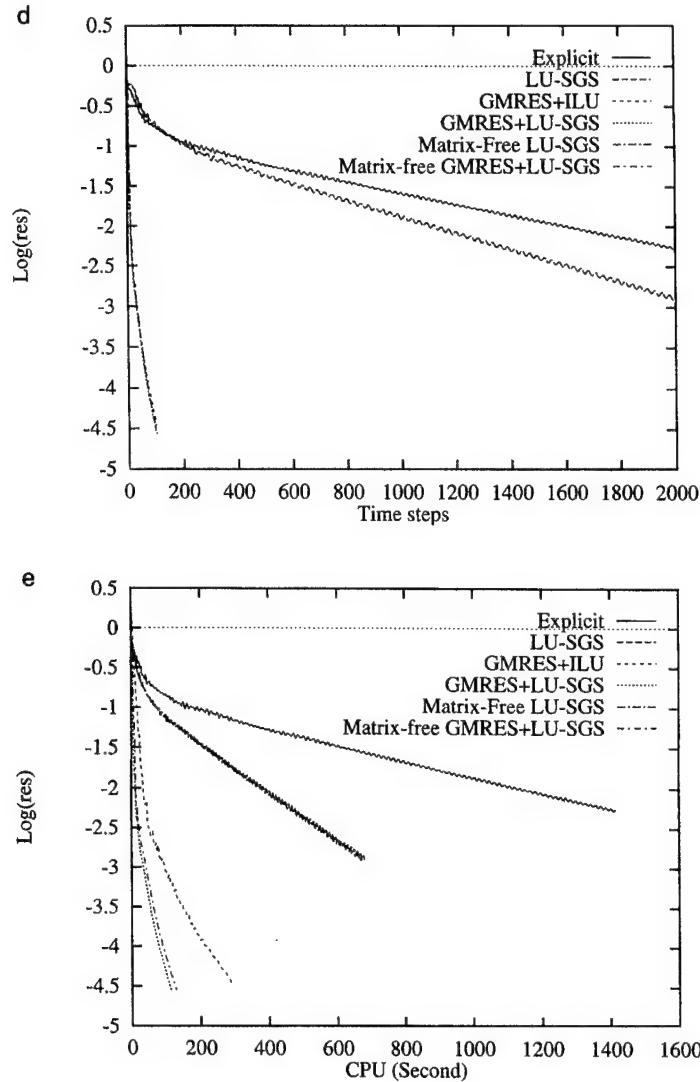
where  $k$  is the number of search directions. Since the GMRES+LU-SGS is completely separated from the flux computation procedure, memory, which is used to compute fluxes can be used by the GMRES+LU-SGS. Overall, the extra storage of the GMRES+LU-SGS method is approximately 10% of the total memory requirements.

## 5. NUMERICAL RESULTS

The present implicit method has been used to compute a variety of compressible flow problems for a wide range of flow conditions, from subsonic to supersonic, for both inviscid and viscous flows, in both 2D and 3D. Only a few typical examples in 3D are presented here to demonstrate the effectiveness and robustness of the present implicit method over the



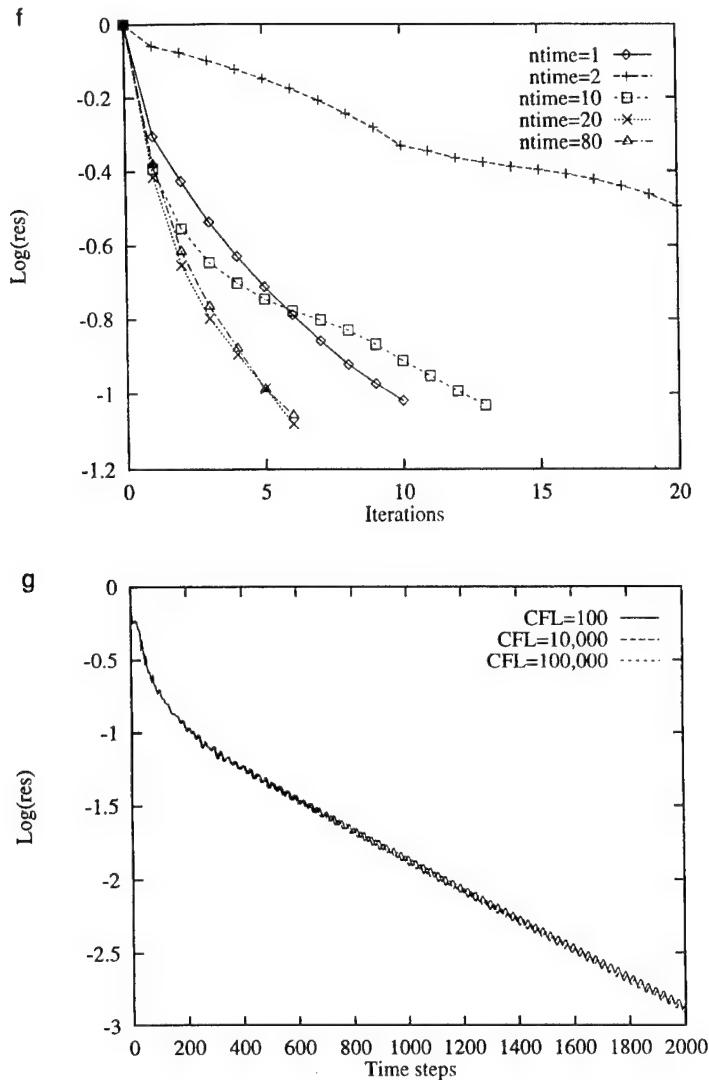
**FIG. 1.** (a) Surface mesh used for computing channel flow (nelem = 68,097, npoin = 13,091, nboun = 4,442). (b) Computed pressure contours on the channel surface at  $M_{in} = 0.5$ . (c) Computed Mach number contours on the channel surface at  $M_{in} = 0.5$ .



**FIG. 1.** (d) Convergence history versus time steps for subsonic channel flow using different schemes: explicit, GMRES+ILU, LU-SGS, GMRES+LU-SGS, matrix-free LU-SGS, and matrix-free GMRES+LU-SGS. (e) Convergence history versus CPU time for subsonic channel flow using different schemes: explicit, GMRES+ILU, LU-SGS, GMRES+LU-SGS, matrix-free LU-SGS, and matrix-free GMRES+LU-SGS.

existing implicit methods. No attempt has been made to use our GMRES+ILU method [6] to solve the Navier–Stokes equations and large-scale problems, as its storage requirement exceeds the memory limitation of the computer available to us.

All of the grids used here were generated by the advancing front technique [17]. All computations were started with uniform flow. The relative  $L_2$  norm of the density residual is taken as a criterion to test convergence history. The solution tolerance for GMRES is set to 0.1 with 10 search directions and 20 iterations. We observed that during the first few time steps, more iterations are spent to solve the system of the linear equations: even 20 iterations cannot guarantee that the stopping criterion will be satisfied for some problems. However, it only takes four or five iterations to solve the linear equations at a later time,



**FIG. 1.** (f) Convergence history of linear system at different time-steps for subsonic flow by matrix-free GMRES+LU-SGS method. (g) Effect of CFL number on convergence history by matrix-free LU-SGS method for subsonic flow.

and global convergence is not affected by a lack of linear system convergence during the first few time steps.

#### A. Inviscid Subsonic Flow in a Channel with a Circular Bump on the Lower Wall

The first example is the well-known Ni's test case: a subsonic flow in a channel with a 10% thick circular bump on the bottom. The length of the channel is 3, its height is 1, and its width is 0.5. The inlet Mach number is 0.5. This is a three-dimensional simulation of a two-dimensional flow. Since no shock waves are present in the flow fields, all solutions were obtained using a second-order scheme without any limiters. The mesh, which contains 13,891 grid points, 68,097 elements, and 4442 boundary points is depicted in Fig. 1a. The

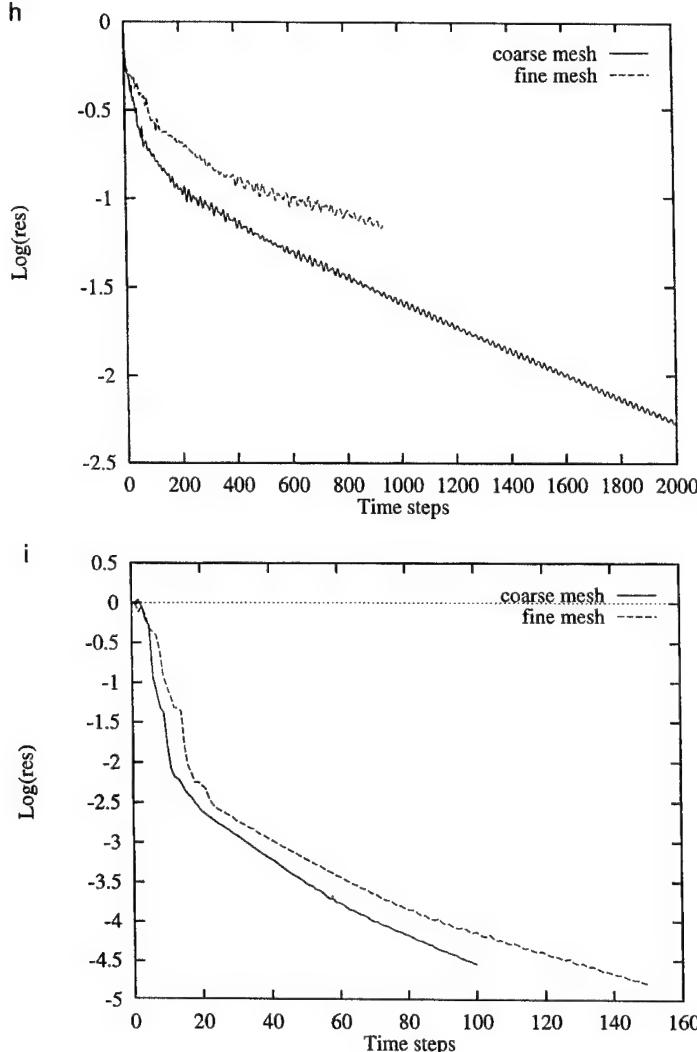


FIG. 1. (h) Convergence history of the residual for coarse and fine meshes for subsonic flow by explicit method. (i) Convergence history of the residual for coarse and fine meshes for subsonic flow by GMRES+LU-SGS implicit method.

computed Mach number and pressure contours in the flow field are shown in Figs. 1b and 1c, respectively. Figures 1d and 1e display a comparison of convergence histories among the explicit scheme, the GMRES+ILU scheme, the LU-SGS scheme, the GMRES+LU-SGS scheme, the matrix-free LU-SGS scheme, and the matrix-free GMRES+LU-SGS scheme versus time steps and CPU time, respectively. The explicit method used a three-stage Runge-Kutta time-stepping scheme with local time stepping and implicit residual smoothing. The computation was advanced with a CFL number of 4. A CFL number of 10,000 was used by all implicit methods in the computation. It is clear that GMRES+LU-SGS methods are superior to both GMRES+ILU and LU-SGS methods. The present GMRES+LU-SGS method is over 100 times faster than its explicit counterpart for this particular case. This is due to the fact that the convergence of the explicit method deteriorates dramatically for

a low-speed flow problem. From Fig. 1e we observe that both the matrix-free LU-SGS scheme and the matrix-free GMRES+LU-SGS scheme yield a convergence history that is identical to their respective matrix counterparts. This indicates that both matrix-free schemes yield solutions that are identical to their matrix counterparts. However, the matrix-free GMRES+LU-SGS scheme is slightly more expensive than its matrix counterpart, as we can see from Fig. 1f. This is due the fact that for each GMRES iteration, the former involves the numerical flux computation, while the latter involves the computation of a matrix vector product, which is apparently cheaper to calculate. It is worth noting that per time step the present LU-SGS method costs approximately half of a three-stage Runge-Kutta explicit method with a residual smoothing. However, the cost of the GMRES+LU-SGS method per time step is not fixed, as more iterations and, therefore, more CPU time, are required to solve the linear system at earlier time steps. This is shown in Fig. 1f, where the convergence history of the linear system at different time steps using the matrix-free GMRES+LU-SGS method is displayed. Typically, only a few iterations are sufficient to meet the stop criterion at latter time-steps. Finally, Fig. 1g illustrates the convergence history of matrix-free LU-SGS method for different CFL number. Although, the LU-SGS method is stable for a

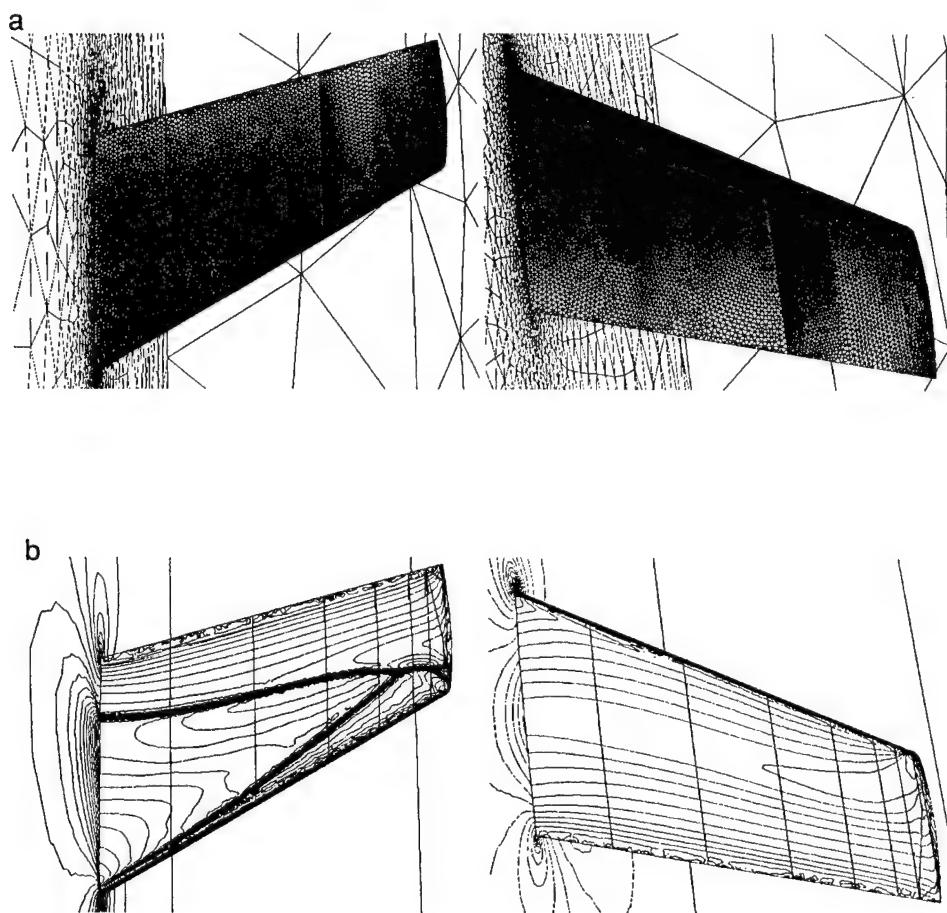


FIG. 2. (a) Upper and lower surface mesh used for M6wing configuration (nelem = 741,098, npoin = 136,051, nboun = 20,762). (b) Computed pressure contours on the upper and lower surface at  $M_{\infty} = 0.84$  and  $\alpha = 3.06^\circ$ .

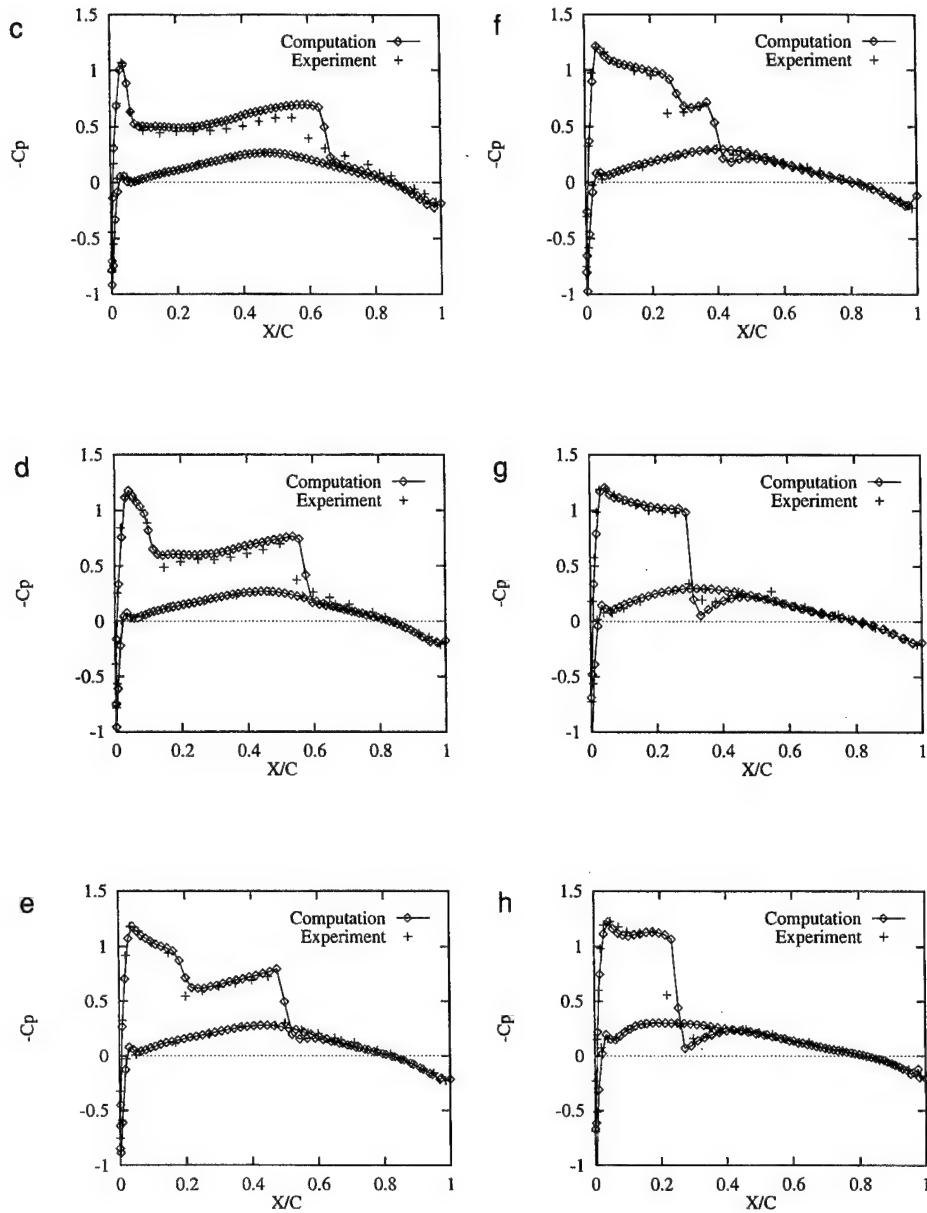
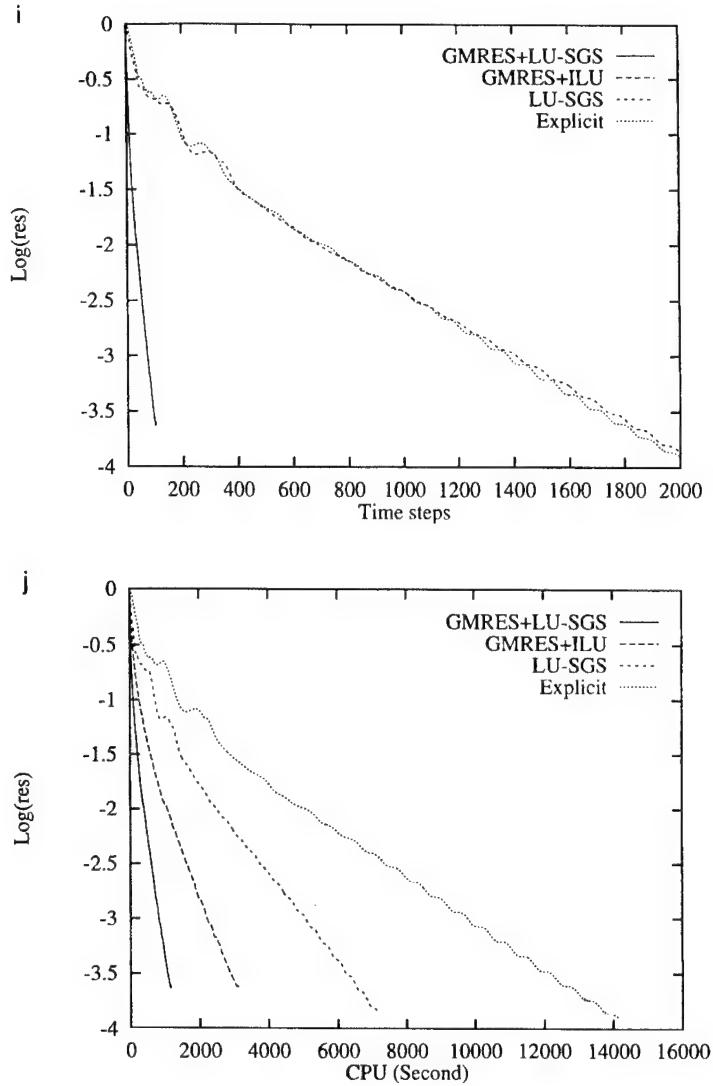


FIG. 2. (c) Comparison between computed and experimental surface pressure coefficient for wing section at 20% semispan. (d) Comparison between computed and experimental surface pressure coefficient for wing section at 44% semispan. (e) Comparison between computed and experimental surface pressure coefficient for wing section at 65% semispan. (f) Comparison between computed and experimental surface pressure coefficient for wing section at 80% semispan. (g) Comparison between computed and experimental surface pressure coefficient for wing section at 90% semispan. (h) Comparison between computed and experimental surface pressure coefficient for wing section at 95% semispan.



**FIG. 2.** (i) Residual convergence history versus time steps for M6wing using different schemes: explicit, matrix-free LU-SGS, GMRES+ILU, and matrix-free GMRES+LU-SGS. (j) Residual convergence history versus CPU time for M6wing using different schemes: explicit, matrix-free LU-SGS, GMRES+ILU, and matrix-free GMRES+LU-SGS.

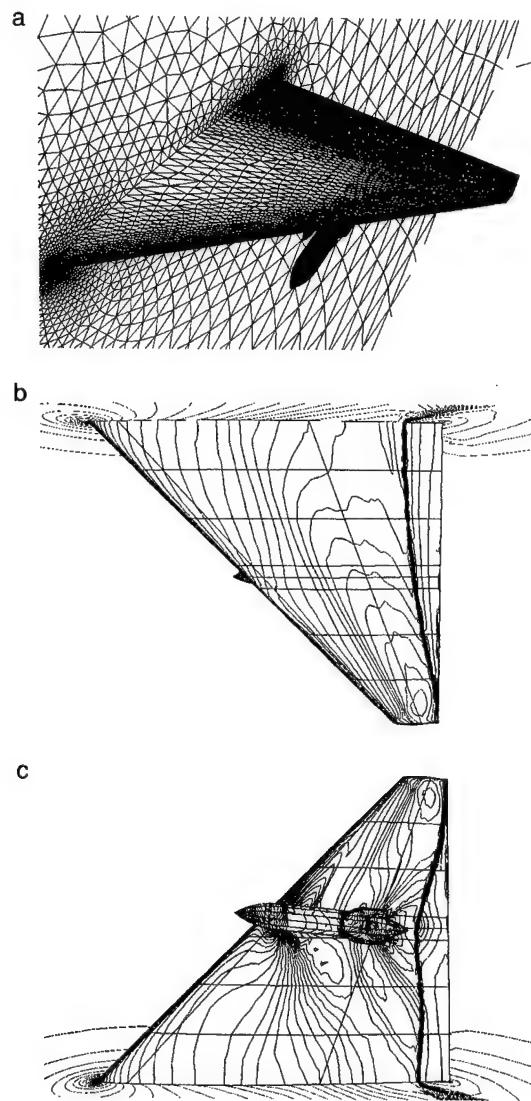
very large CFL number, the convergence history is almost indistinguishable for all the three CFL numbers used.

Finally, the same computation has been performed on a finer mesh to study the behavior of convergence history as the grid is refined. The refined mesh contains 99,683 grid points, 533,060 elements, and 17,391 boundary points. Figures 1h and 1i show the convergence histories of the residual for coarse and finer grids using the explicit method and the matrix-free GMRES+LU-SGS method, respectively. As expected, the explicit method for the refined mesh requires approximately twice the number of time steps to achieve the same convergence; the behavior of convergence history for the implicit method is quite similar

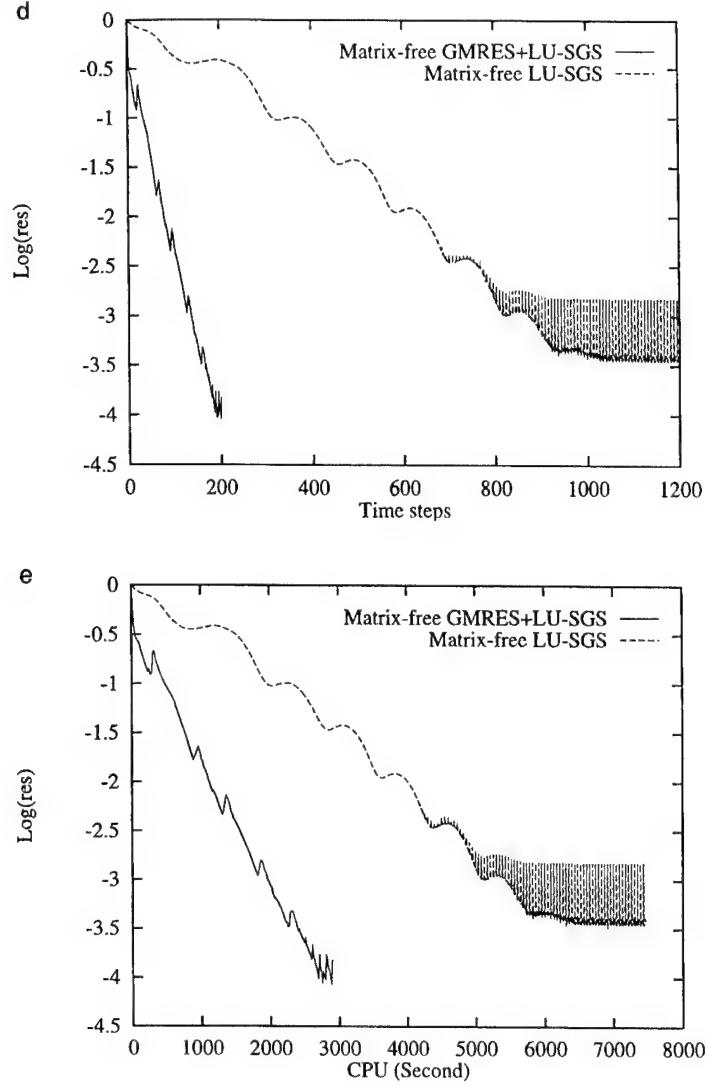
as on a coarse mesh, although the rate of convergence does slow down, demonstrating that the advantage of the present implicit method is not diminished for finer grids.

### B. ONERA M6 Wing Configuration

The second, well-documented case is the inviscid transonic flow over a ONERA M6 wing configuration. The M6 wing has a leading-edge sweep angle of  $30^\circ$ , an aspect of 3.8, and a taper ratio of 0.562. The airfoil section of the wing is the ONERA "D" airfoil, which is a 10% maximum thickness-to-chord ratio conventional section. The flow solutions are presented at a Mach number of 0.84 and an angle of attack of 3.06. The mesh used in the computation

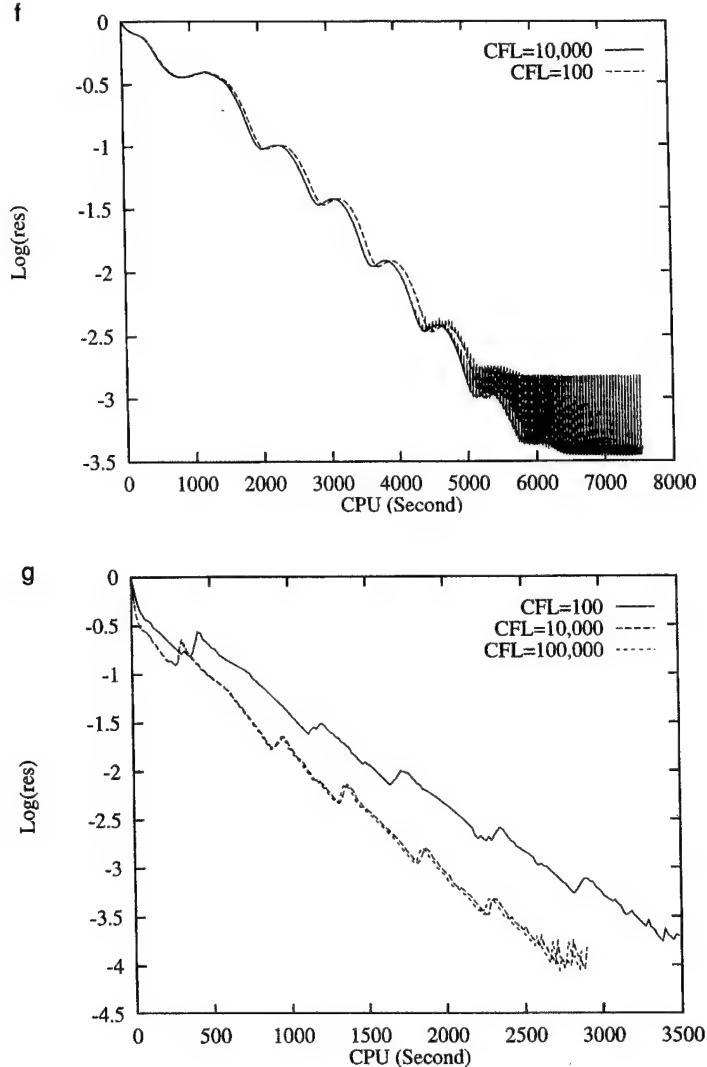


**FIG. 3.** (a) Surface mesh used for Wing/Pylon/Finned-Store configuration (nelem = 1,329,694, npoin = 239,547, nboun = 27,359). (b) Computed pressure contours on the upper surface at  $M_\infty = 0.95$  and  $\alpha = 0^\circ$ . (c) Computed pressure contours on the lower surface at  $M_\infty = 0.95$  and  $\alpha = 0^\circ$ .



**FIG. 3.** (d) Residual convergence history versus time steps for Wing/Pylon/Finned-Store configuration using matrix-free LU-SGS and matrix-free GMRES+LU-SGS. (e) Residual convergence history versus CPU time for Wing/Pylon/Finned-Store configuration using different schemes: matrix-free LU-SGS and matrix-free GMRES+LU-SGS.

consists of 741,095 elements, 136,051 grid points, and 20,762 boundary points. The upper and surface meshes are shown in Fig. 2a. The computed pressure contours on the upper and lower surfaces are displayed in Fig. 2b. The upper surface contours clearly show the sharply captured lambda-type shock structure formed by the two inboard shock waves, which merge together near 80% semispan to form the single strong shock wave in the outboard region of the wing. The computed pressure coefficient distributions are compared with experimental data [18] at six spanwise stations in Figs. 2c–2h. We can observe that there is only one grid point within the shock structure; this demonstrates the sharp shock-capturing ability of AUSM+ scheme. The results obtained compare closely with experimental data, except at the root stations, due to lack of viscous effects. Figures 2i and 2j display a comparison

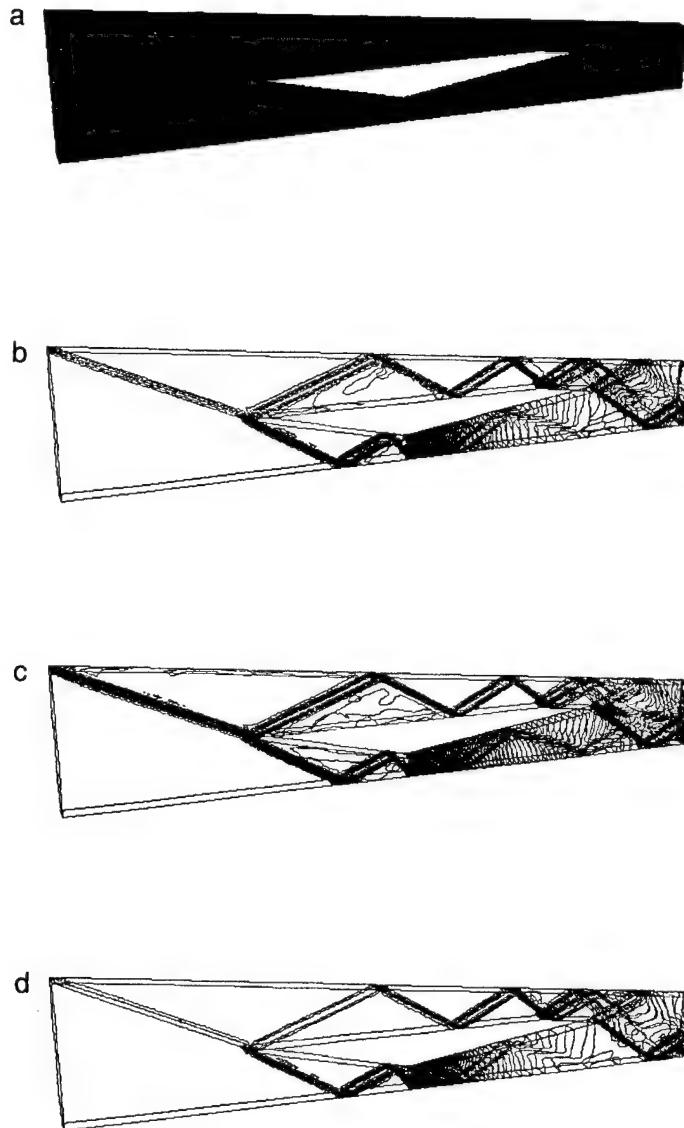


**FIG. 3.** (f) Effect of CFL number on convergence history for for Wing/Pylon/Finned-Store configuration using matrix-free LU-SGS. (g) Effect of CFL number on convergence history for for Wing/Pylon/Finned-Store configuration using matrix-free GMRES+LU-SGS.

of convergence histories among the explicit scheme, matrix-free LU-SGS scheme, GMRES+ILU scheme, and matrix-free GMRES+LU-SGS scheme versus time steps and CPU time, respectively. GMRES+LU-SGS methods provide the best convergence performance. The present GMRES+LU-SGS method is about three times faster than the GMRES+ILU method, about six times faster than the LU-SGS methods and 14 times faster than its explicit method.

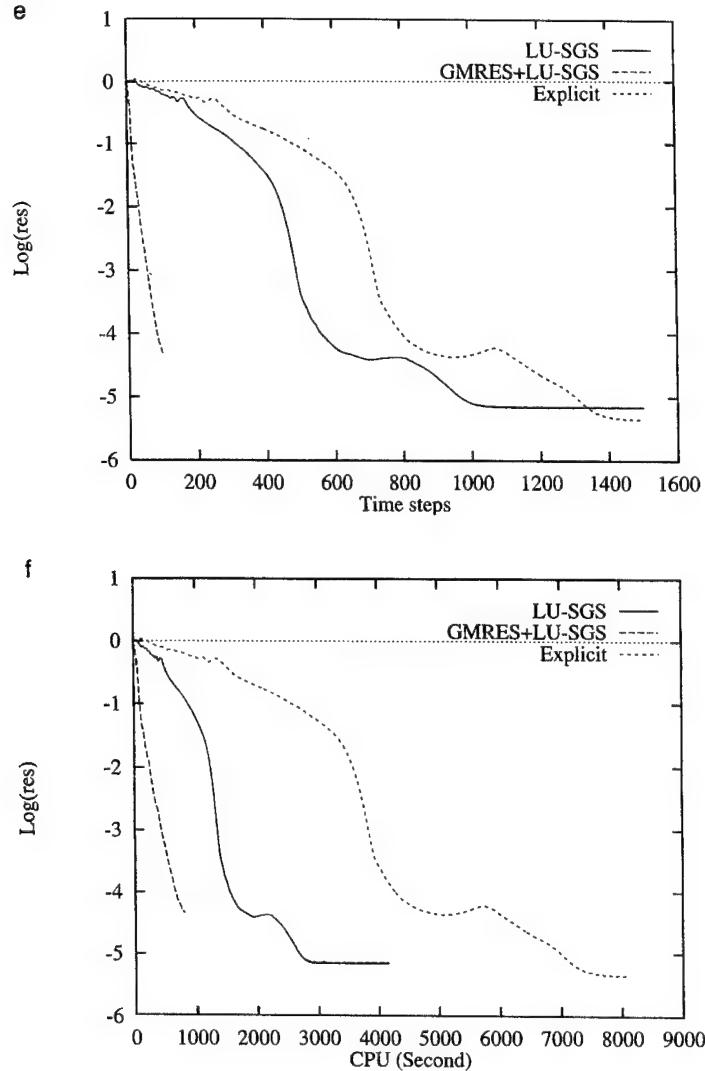
### C. Wing/Pylon/Finned-Store Configuration

The third test case is conducted for a wing/pylon/finned-store configuration reported in Ref. [19]. The configuration consists of a clipped delta wing with a  $45^\circ$  sweep composed



**FIG. 4.** (a) Surface mesh used for computing internal supersonic flow (nelem = 542,895, npoin = 106,285, nboun = 30,094). (b) Computed density contours for supersonic inlet flow ( $M_{in} = 3$ ). (c) Computed Mach number contours for supersonic inlet flow ( $M_{in} = 3$ ). (d) Computed pressure contours for supersonic inlet flow ( $M_{in} = 3$ ).

of a constant NACA64010 symmetric airfoil section. The wing has a root chord of 16 in., a semispan of 13 in., and a taper ratio of 0.134. The pylon is located at the midspan station and has a cross section characterized by a flat plate closed at the leading and trailing edges by a symmetrical ogive shape. The width of the pylon is 0.294 in. The four fins on the store are defined by a constant NACA0008 airfoil section with a leading-edge sweep of  $45^\circ$  and a truncated tip. The mesh used in the computation is shown in Fig. 3a. It contains 1,329,694 elements, 239,547 grid points, and 27,359 boundary points. The flow solutions are presented at a Mach number of 0.95 and an angle of attack of  $0^\circ$ . Figures 3b and 3c show the pressure contours on the upper and lower wing surface, respectively. Because of large size of mesh, no attempt has been made using matrix LU-SGS and GMRES+LU-SGS methods to compute

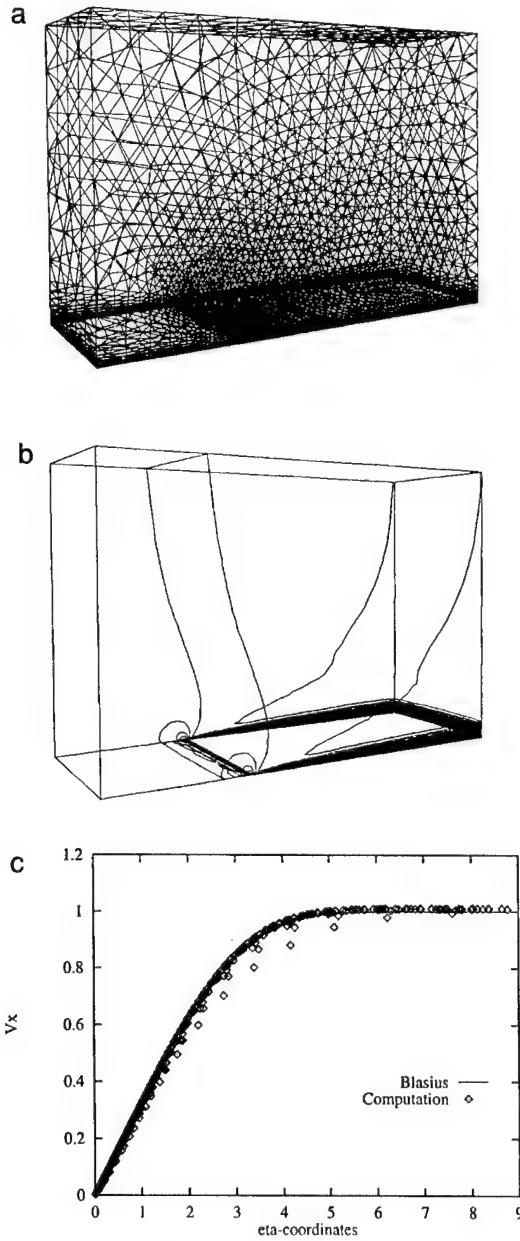


**FIG. 4.** (e) Residual convergence history versus time steps for supersonic inlet configuration using different schemes: matrix-free LU-SGS, matrix-free GMRES+LU-SGS, and explicit method. (f) Residual convergence history versus CPU time for supersonic inlet configuration using different schemes: matrix-free LU-SGS, matrix-free GMRES+LU-SGS, and explicit method.

this problem. Figures 3d and 3e display a comparison of convergence histories between a matrix-free LU-SGS scheme and a matrix-free GMRES+LU-SGS scheme versus time steps and CPU time, respectively. Again, the GMRES+LU-SGS method provides faster convergence than the LU-SGS method. Figures 3f and 3g illustrate the convergence history using different CFL numbers of GMRES+LU-SGS and LU-SGS methods, respectively. Again, we can see that, although the LU-SGS method is stable for a very large CFL number, the convergence history is almost indistinguishable for all the three CFL numbers used. However, a substantial gain can be achieved by using a larger CFL number for GMRES+LU-SGS method, although no significant difference can be observed once the CFL number is large enough.

#### D. Supersonic Duct Flow

This internal inviscid supersonic flow case, taken from Nakahashi and Saito [20], represents part of a scramjet intake. The inlet Mach number is 3. The total length of the device is  $l = 8.0$ , and the element size was set uniformly throughout the domain to  $\delta = 0.03$ . The mesh shown in Fig. 4a consists of 542,895 elements, 106,285 grid points, and 30,094 boundary points. The computed density, Mach numbers, and pressure contours are shown in Figs. 4b,



**FIG. 5.** (a) Surface mesh used for flat plate configuration (nelem = 81,885, npoin = 15,694, nboun = 3,774). (b) Computed Mach number contours for flat plate at  $M_\infty = 0.4$ ,  $\alpha = 0.0$ , and  $Re = 10,000$ . (c) Computed boundary layer velocity profile over a flat plate at  $M_\infty = 0.4$ ,  $\alpha = 0.0$ , and  $Re = 10,000$ .

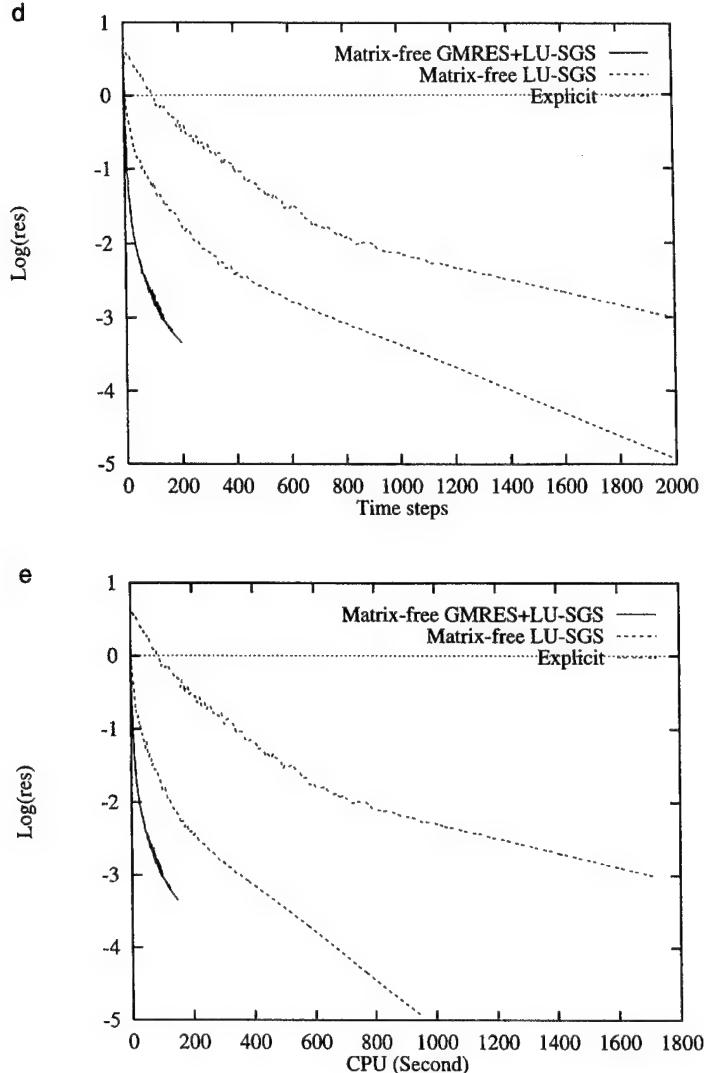


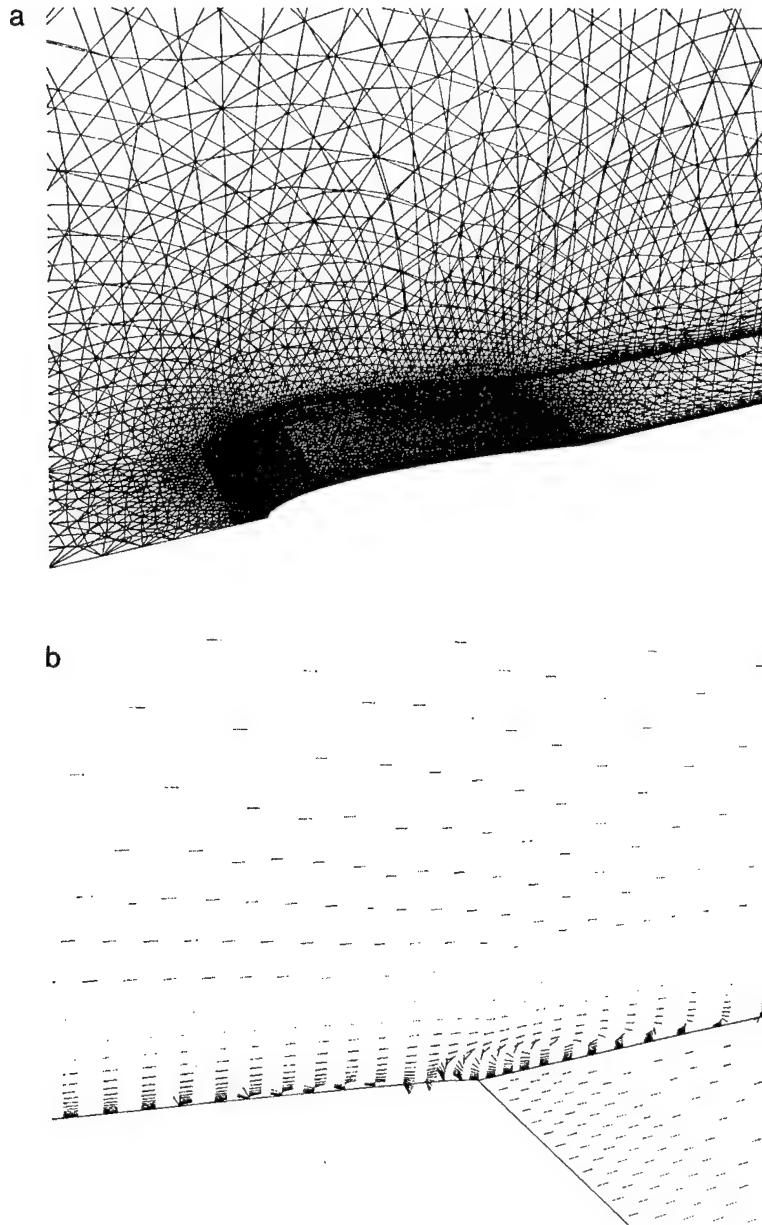
FIG. 5. (d) Residual convergence history versus time steps for flat plate using different schemes: explicit, matrix-free LU-SGS, and matrix-free GMRES+LU-SGS. (e) Residual convergence history versus cpu time for flat plate using different schemes: explicit, matrix-free LU-SGS, and matrix-free GMRES+LU-SGS.

4c, and 4d, respectively. Figures 4e and 4f illustrate the convergence history among different numerical schemes: matrix-free LU-SGS, matrix-free GMRES+LU-SGS, and explicit methods, respectively. It indicates that the GMRES+LU-SGS method is superior to the LU-SGS method. CPU time comparison shows that the GMRES+LU-SGS method is about eight time faster than the explicit method for this particular problem.

#### E. Laminar Flow Past a Flat Plate

In this test case, Blasius boundary layers are computed for a flat plate at a Mach number of 0.4 and a chord Reynolds number of 10,000. The computational domain is considered from  $x = -0.5$  to  $x = 1$ ,  $y = 0$  to  $y = 1$ , and  $z = 0$  to  $z = 0.5$ , where the plate starts at  $x = 0$ .

The mesh used in the computation is shown in Fig. 5a. It contains 81,885 elements, 15,694 points, and 3774 boundary points. The computed Mach number contours in the flow field are depicted in Fig. 5b, where the development of a boundary layer can be clearly observed. Figure 5c shows the comparison of the Blasius velocity profile and the computed velocity profiles as scaled by the Blasius similarity law for all boundary layer points. The Blasius velocity profile is almost identically matched by all data points with the exception of a



**FIG. 6.** (a) Surface mesh used for NACA0012 airfoil configuration (nelem = 697,655, npoin = 128,448, nboun = 22,925). (b) Computed velocity vector distribution near leading edge of airfoil at  $M_\infty = 0.5$ ,  $\alpha = 0.0$ , and  $Re = 5,000$ .

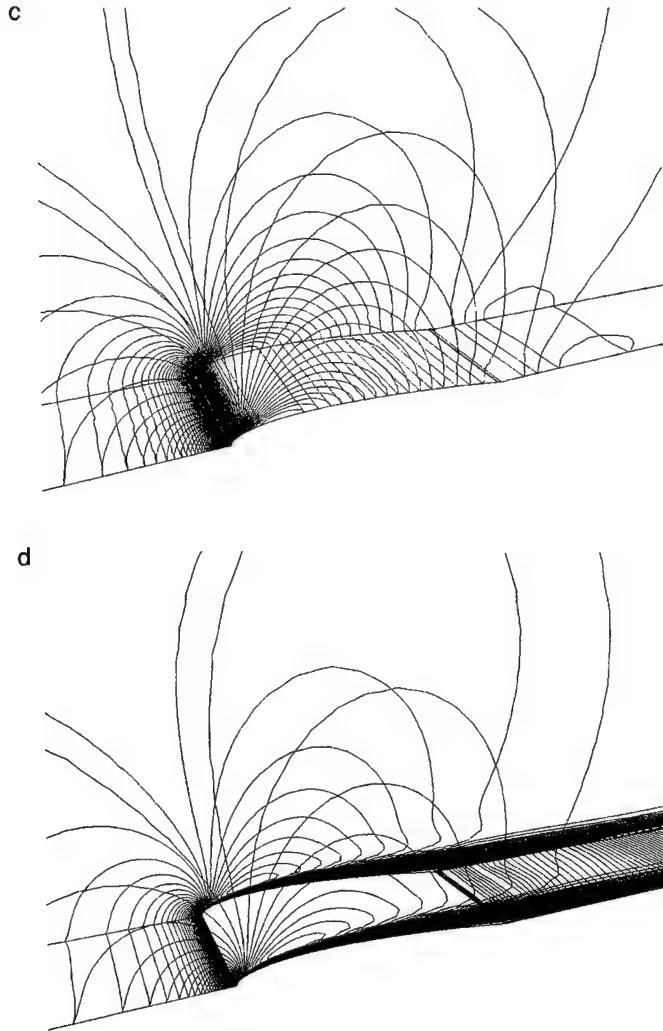
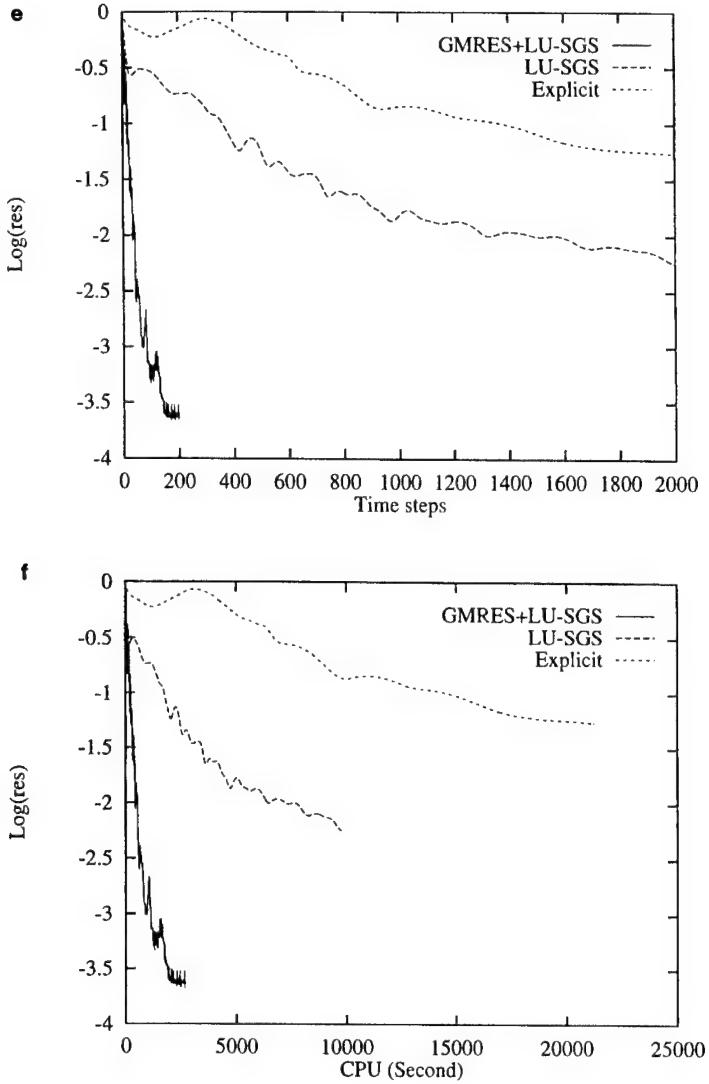


FIG. 6. (c) Computed pressure contours on the surface at  $M_\infty = 0.5$ ,  $\alpha = 0.0$ , and  $Re = 5,000$ . (d) Computed Mach number contours on the surface at  $M_\infty = 0.5$ ,  $\alpha = 0.0$ , and  $Re = 5,000$ .

few points near leading edge. The slight discrepancy for these points is attributed to the leading edge singularity. Figures 5d and 5e show a comparison of convergence histories among different numerical schemes: matrix-free LU-SGS, matrix-free GMRES+LU-SGS, and explicit methods, respectively. It indicates that the GMRES+LU-SGS method is superior to the LU-SGS method. CPU time comparison shows that the GMRES+LU-SGS method is about 10 times faster than the explicit method for this particular problem.

#### F. Laminar Flow over a NACA0012 Airfoil

This test case involves a laminar flow past a NACA0012 airfoil at a Mach number of 0.5, an angle of attack of  $0^\circ$ , and a chord Reynolds number of 5000. This computation was performed to see the effectiveness of the present matrix-free GMRES+LU-SGS method for the solution of the Navier-Stokes equations. The mesh used in the computation is shown in Fig. 6a. It contains 697,655 elements, 128,488 grid points, and 22,925 boundary points.



**FIG. 6.** (e) Residual convergence history versus time steps for NACA0012 airfoil using different schemes: explicit, matrix-free LU-SGS, and matrix-free GMRES+LU-SGS. (f) Residual convergence history versus cpu time for NACA0012 airfoil using different schemes: explicit, matrix-free LU-SGS, and matrix-free GMRES+LU-SGS.

The computed velocity vector distribution in the vicinity of the trailing edge of the airfoil is shown in Fig. 6b, where the separation and a small recirculation bubble can be clearly observed. The computed separation point is at 81.6% chord, which compares well to the one obtained by Mavriplis [22]. The computed pressure and Mach-number contours are shown in Figs. 6c and 6d, respectively. Figures 6e and 6f illustrate the convergence history among different numerical schemes: matrix-free LU-SGS, matrix-free GMRES+LU-SGS, and explicit methods, respectively. It indicates that the GMRES+LU-SGS method is far superior to the LU-SGS method. CPU time comparison shows that the GMRES+LU-SGS method is more than two orders of magnitude faster than the explicit method for this particular problem. The effectiveness of the present matrix-free GMRES+LU-SGS method for the solution of the Navier-Stokes equations is clearly demonstrated in this example.

## 6. CONCLUSIONS

A matrix-free implicit method has been developed to solve the three-dimensional Navier-Stokes and Euler equations on unstructured meshes. The developed method has been used to compute the compressible flows around 3D complex aerodynamic configurations for a wide range of flow conditions from subsonic to supersonic. The numerical results obtained indicate that the use of the GMRES+LU-SGS method leads to a significant increase in performance over the best current implicit methods, the GMRES+ILU and the LU-SGS methods, while maintaining memory requirements that are competitive with its explicit counterpart. In comparison to the explicit method, we demonstrate an overall speedup factor from eight to more than one order of magnitude for all test cases. The GMRES+LU-SGS method has also been extended and applied successfully to solve the unsteady Euler and Navier-Stokes equations and will be reported in a later paper. The current work is to extend the present GMRES+LU-SGS method for turbulent flow problems.

## ACKNOWLEDGMENTS

This research was sponsored by the Defense Special Weapon Agency. Dr. Michael E. Giltrud served as the technical program monitor. Partial funding for the last author was also provided by the Air Force Office of Scientific Research. Dr. Leonidas Sakell served as the technical monitor.

## REFERENCES

1. B. Stoerfflet, Implicit finite element methods for the Euler equations, in *Numerical Methods for the Euler Equations of Fluid Dynamics*, edited by F. Angrand (SIAM, Philadelphia, 1985).
2. J. T. Batina, Implicit flux-split Euler schemes for unsteady aerodynamic analysis involving unstructured dynamic meshes, *AIAA J.* **29**(11), (1991).
3. V. Venkatakrishnan and D. J. Mavriplis, Implicit solvers for unstructured meshes, *J. Comput. Phys.* **105**, 83 (1993).
4. D. D. Knight, *A Fully Implicit Navier-Stokes Algorithm Using an Unstructured Grid and Flux Difference Splitting*, AIAA Paper 93-0875, 1993.
5. D. L. Whitaker, *Three-dimensional Unstructured Grid Euler Computations Using a Fully-Implicit, Upwind Method*, AIAA Paper 93-3337, 1993.
6. H. Luo, J. D. Baum, R. Löhner, and J. Cabello, *Implicit Schemes and Boundary Conditions for Compressible Flows on Unstructured Meshes*, AIAA Paper 94-0816, 1994.
7. T. J. Barth and S. W. Linton, *An Unstructured Mesh Newton Solver for Compressible Fluid Flow and Its Parallel Implementation*, AIAA Paper 95-0221, 1995.
8. Y. Saad, Iterative methods for sparse linear systems, on unstructured meshes, 1996.
9. Y. Saad and M. H. Schultz, GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems, *SIAM J. Sci. Stat. Comp.* **7**(3), 89 (1988).
10. A. Jameson and S. Yoon, Lower-upper implicit schemes with multiple grids for the Euler equations, *AIAA J.* **25**(7), (1987).
11. M. Soetrisno, S. T. Imlay, and D. W. Roberts, *A Zonal Implicit Procedure for Hybrid Structured-Unstructured Grids*, AIAA Paper 94-0617, 1994.
12. I. Men'shov and Y. Nakamura, An implicit advection upwind splitting scheme for hypersonic air flows in thermochemical nonequilibrium, in *6th Int. Sympos. on CFD*, 1995.
13. D. Sharov and K. Nakahashi, *Reordering of 3-D Hybrid Unstructured Grids for Vectorized LU-SGS Navier-Stokes Computations*, AIAA Paper 97-2102, 1997.
14. M. S. Liou, *Progress towards an Improved CFD Method: AUSM+*, AIAA Paper 95-1701, Jun. 1995.

15. B. Van Leer, Towards the ultimate conservative difference scheme. II. Monotonicity and conservation combined in a second-order scheme, *J. Comput. Phys.* **14**, 361 (1974).
16. P. L. Roe, Approximate Riemann solvers, parameter vectors and difference schemes, *J. Comput. Phys.* **43**, 357 (1981).
17. J. D. Löhner and P. Parikh, Three-dimensional grid generation by the advancing front method, *Int. J. Numer. Methods Fluids* **8**, 1135 (1988).
18. V. Schmitt and F. Charpin, Pressure distributions on the ONERA M6-wing at transonic Mach numbers, *Experiment Data Base for Computer Program Assessment*, AGARD AR-138, 1979.
19. E. R. Illeim, *CFD Wing/Pylon/Finned Store Mutual Interference Wind Tunnel Experiment*, AEDC-TSR-91-P4, Arnold Engineering Development Center, Arnold AFB, TN, Jan. 1991.
20. K. Nakahashi and E. Saitoh, *Space-Marching Method on Unstructured Grid for Supersonic Flows with Embedded Subsonic Regions*, AIAA-96-0418, 1996.
21. H. Luo, J. D. Baum, and R. Löhner, An edge-based upwind finite element scheme for the Euler equations, *AIAA J.* **32**(6), (1994).
22. D. J. Mavriplis and A. Jameson, Multigrid solution of the Navier–Stokes equations on triangular meshes, *AIAA J.* **28**(8), (1990).

## APPENDIX 2: PARALLEL GRID GENERATION



**AIAA-00-1005**

**A Parallel Advancing Front  
Grid Generation Scheme**

**Rainald Löhner**

George Mason University, Fairfax, VA

**38th Aerospace Sciences  
Meeting & Exhibit**

**10-13 January 2000 / Reno, NV**

# A PARALLEL ADVANCING FRONT GRID GENERATION SCHEME

Rainald Löhner

Institute for Computational Science and Informatics  
 M.S. 4C7, George Mason University  
 Fairfax, VA 22030-4444, USA

## ABSTRACT

A parallel advancing front scheme has been developed. The domain to be gridded is first subdivided spatially using a relatively coarse octree. Boxes are then identified and gridded in parallel. A scheme that resembles closely the advancing front technique on scalar machines is recovered by only considering the boxes of the active front that generate small elements. The procedure has been implemented on the SGI Origin class of machines using the shared memory paradigm. Timings for a variety of cases show speedups similar to those obtained for flow codes. The procedure has been used to generate grids in excess of a hundred million elements.

**Keywords.** Unstructured Grid Generation, Parallel Computing, CFD.

## 1. INTRODUCTION

The widespread availability of parallel machines with large memory, solvers that can harness the power of these machines, and the desire to model in ever increasing detail geometrical and physical features has lead to a steady increase in the number of points used in field solvers. Grids in excess of  $10^7$  elements have become common for production runs in Computational Fluid Dynamics (CFD) [Bau93, Bau95, Jou98, Yos98, Mav99] and Computational Electromagnetics (CEM) [Dar97, Mor97]. The expectation is that in the near future grids in excess of  $10^8 - 10^9$  elements will be required. While many solvers have been ported to parallel machines, grid generators have lagged behind. For applications where remeshing is an integral part of simulations, e.g. problems with moving bodies [Löh90, Mes93, Mes95, Bau96, Kam96, Löh98a, Has98] or changing topologies [Bau98, Bau99], the time required for mesh regeneration can easily consume more than 50% of the total time required to solve the problem. Faced with this situation, a number of efforts have been reported on parallel grid generation [Löh92, dCo94, Sho95, dCo95, Oku96, Che97, Oku97, Sai99].

The two most common ways of generating unstructured grids are the Advancing Front Technique (AFT) [Per87, Per88, Löh88a,b, Per90, Per92, Jin93, Fry94, Löh96] and the Generalized Delaunay Triangulation (GDT) [Bak89, Geo91, Wea92, Wea94, Mar95]. The AFT introduces one element at a time, while the

GDT introduces a new point at a time. Thus, both of these techniques are, in principle, scalar by nature, with a large variation in the number of operations required to introduce a new element or point. While coding and data structures may influence the scalar speed of the 'core' AFT or GDT, one often finds that for large-scale applications, the evaluation of the desired element size and shape in space, given by background grids, sources or other means [Löh96] consumes the largest fraction of the total grid generation time. Unstructured grid generators based on the AFT may be parallelized by invoking distance arguments, i.e. the introduction of a new element only affects (and is affected by) the immediate vicinity. This allows for the introduction of elements in parallel, provided that sufficient distance lies between them.

Several years ago, the author and his colleagues introduced a parallel AFT based on the subdivision of the background grid [Löh92, Sho95]. While used for some demonstration runs, this scheme was not general enough for a production environment. The background grid had to be adapted in order to be sufficiently fine for a balanced workload. As only background grid elements covering the domain to be gridded were allowed, complex in/out tests had to be carried out to remove refined elements lying outside the domain to be gridded. Furthermore, element size specified at CAD entities could not be 'propagated' into the domain, as is the case in the scalar AFT, disabling an option favoured by many users and rendering many grid generation data sets unusable. The otherwise positive experience gained with this parallel AFT prompted the search for a more general parallel AFT. The key requirement was a parallel AFT

that changes the current, evolved and mature scalar AFT as little as possible, while achieving significant speedups on common parallel machines. This implies that the parallelism should be applied at the level of the current front, and not globally.

## 2. PARALLEL GRIDDING SCHEME

The advancing front technique attempts to introduce an element at a time into an as yet ungridded domain by eliminating the face generating the smallest new element from the front [Löh88a,b]. Given that the introduction of an element only affects its immediate neighbourhood, one could, in principle, introduce many elements at the same time, provided they are sufficiently far apart. A convenient way of delimiting the possible zones where elements may be introduced by each processor is via boxes. These boxes may be obtained in a variety of ways, i.e. via bins, binary recursive trees, or octrees. We have found the octree to be the best of these possibilities, particularly for grids with a large variation of element size. In order to recover a parallel gridding procedure that resembles closely the advancing front technique on scalar machines, only the boxes covering the active front in regions where the smallest new elements are being introduced are considered. After these boxes have been filled with elements, the process starts anew: a new octree is built, new boxes are created and meshed in parallel. The procedure is summarized schematically for a 2-D case in Figure 1.

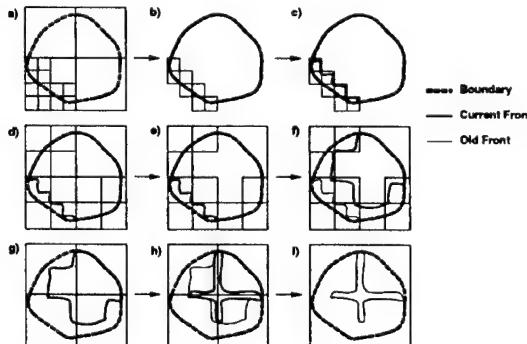


Figure 1 Parallel Grid Generation

At the end of each parallel gridding pass, each one of the boxes gridded can have an internal boundary of faces. For a large number of boxes, this could result in a very large number of faces for the active front. This problem can be avoided by shifting the boxes slightly, and then regridding them again in parallel, as shown in Figure 2. This simple technique has the effect of eliminating almost all of the faces between boxes with a minor modification of the basic parallel gridding algorithm.

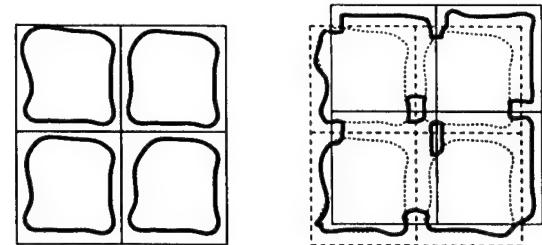


Figure 2 Shift and Regrid Technique

If we define as  $d_{min}$  the minimum element size in the active front, and as  $s_{min}$  the minimum box size in which elements are to be generated, the parallel AFT proceeds as follows:

**WHILE:** There are active faces left:

- Form an octree with minimum octant size  $s_{min}$  for the active points;
- Retain the octants that have faces that will generate elements of size  $d_{min}$  to  $c_l \cdot d_{min}$ ;
- If too many octants are left: agglomerate them into boxes;
- DO ISHFT=0,2:
  - IF: ISHFT.NE.0:
    - Shift the boxes by a preset amount;
  - ENDIF
  - Generate, in parallel, elements in these boxes, allowing only elements up to a size of  $c_l \cdot d_{min}$ ;
- ENDDO
- Increase  $d_{min} = 1.5 * d_{min}$ ,  $s_{min} = 1.5 * s_{min}$ ;

**ENDWHILE**

The increase factor allowed is typically in the range  $c_l = 1.5 - 2.0$ . The shift vectors are given by

$$s = \delta_s(1, 1, 1), \quad \delta_s = \begin{cases} + & \min(0.5 * s_{min}, 2.0 * d_{min}) \\ - & \end{cases}.$$

We remark that the octree used to compute the boxes for parallel grid generation is very coarse compared to the element size specified by the user. The edge-length of the finest octree box is of the order of 20 to 50 times the specified element size. This implies that its construction is very fast, and can be accomplished on a single processor without discernable CPU penalty.

## 3. WORK ESTIMATION AND BALANCE

The procedure outlined above will work optimally if each box requires approximately the same CPU time to complete its grid. This implies that a good work estimate should be provided. Given that the boxes are not body conforming, even for uniform grids the volume to be gridded can vary drastically from box to box. A balanced workload can be obtained by starting

with many boxes, estimating the work to be done for each of them, and then gridding in parallel **groups of boxes** with similar workload.

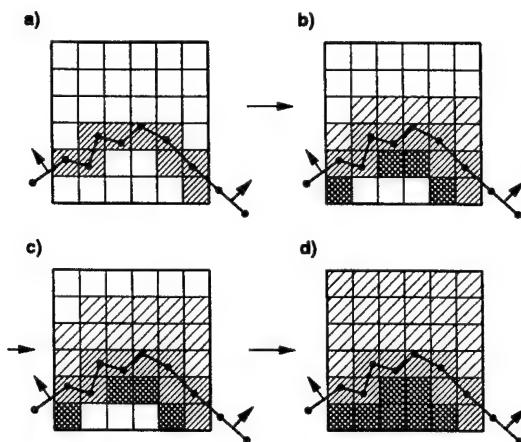


Figure 3 Estimation of Volume to be Gridded

The volume to be gridded is estimated by the marching cubes procedure shown schematically in Figure 3. Given the dimensions of the box, and the list of active faces, the box is first subdivided into voxels (i.e. small cubes). In a first pass over the faces, the voxels cut by faces are marked as 'inside the domain', and an average normal is computed for each cut voxel. This normal information is used in a subsequent pass over the voxels in order to mark the neighbours of cut voxels as either inside or outside the domain to be gridded. The remaining voxels are then marked as inside or outside in several sweeps over the voxels. These sweeps are carried out until no further voxels can be marked. Finally, a work estimate is obtained by summing the expected number of elements in each of the voxels marked as inside the domain to be gridded. This work estimation procedure is done in parallel.

Given the estimated work in each of the boxes, the load is balanced in such a way that each processor receives a similar amount of work. The assumption is made that the number of boxes is always larger than the number of processors. Should this not be the case, the boxes are subdivided further (8 new boxes for each box). If any given box has a work estimate that lies above the average work per processor, this box is also subdivided further. This 'greedy' work balancing algorithm may be summarized as follows:

a) Obtain a minimum nr. of active boxes:

**WHILE:** The number of boxes is smaller than the number of processors:

- Subdivide boxes (1:8);

**ENDWHILE**

b) Balance the work:

**WHILE:** Work unbalanced

- Estimate, in parallel, the work for each box;
- Obtain average work per processor;
- **IF:** A box has a work estimate above average:  
Subdivide it further (1:8) and balance again;
- **ENDIF**
- Attempt to group boxes in such a way that the work in each group is close to average;
- **IF:** Good work balance impossible:  
Subdivide boxes with highest work estimate (1:8) and balance again;
- **ENDIF**

**ENDWHILE**

In many instances, boxes within a group will share a common face. In order to avoid the (unnecessary) buildup of many faces at the borders of boxes, an attempt is made to agglomerate these neighbouring boxes within groups. This is done recursively by checking, for any pair of boxes, if two of the dimensions are the same and the boxes are coincident in the remaining dimension. If so, the boxes are merged. Boxes are merged recursively, until no pair of boxes passes the test outlined above.

#### 4. MESH IMPROVEMENT

After the generation of the mesh using the parallel advancing front technique has been completed, the mesh quality is improved by a combination of several algorithms, such as:

- Diagonal swapping,
- Removal of Bad Elements, and
- Laplacian smoothing.

##### 4.1 Diagonal Swapping

Diagonal swapping attempts to improve the quality of the mesh by reconnecting locally the points in a different way. Examples of possible 3-D swaps are shown in Figures 4,5.

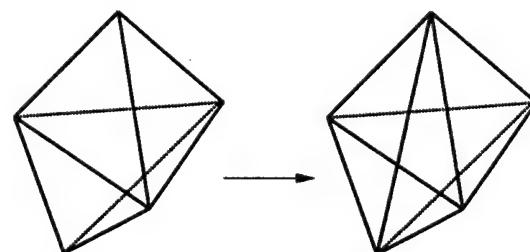


Figure 4 Diagonal Swap Case 2:3

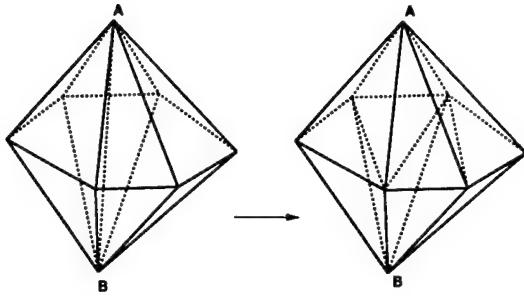


Figure 5 Diagonal Swap Case 6:8

The optimality criterion used is the one proposed by George (1999).

$$Q = \frac{h_{\max} S}{V}$$

where  $h_{\max}$ ,  $S$  and  $V$  denote the maximum edge length, total surface area and volume of a tetrahedron. The number of cases to be tested can grow factorially with the number of elements surrounding an edge. Figure 6 shows the possibilities to be tested for 4, 5 and 6 elements surrounding an edge.

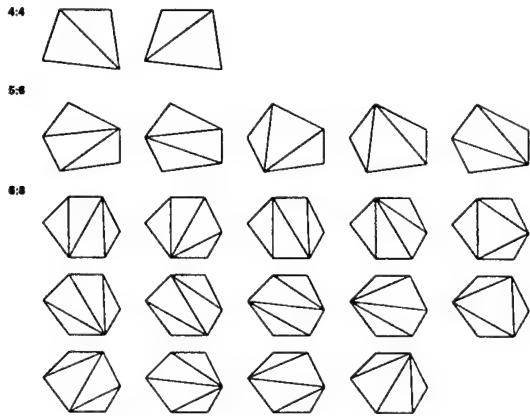


Figure 6 Swapping Cases

Given that these tests are computationally intensive, considerable care is required when coding a fast diagonal swapper. Techniques that were used in the present implementation include:

- Treatment of bad ( $Q > Q_{tol}$ ), untested elements only;
- Processing of elements in an ordered way, starting with the worst (highest chance of reconnection);
- Rejection of bad combinations at the earliest possible indication of worsening quality;
- Marking of tested and unswapped elements in each pass.

As stated before, the computationally intensive part of any diagonal swapping is sifting through all possibilities in order to find a better local point connectivity. One observes that a very large number of tests are

required to obtain an improvement. This implies that parallelizing only the checking portion of a diagonal swapper, which can easily be done as no swapping actually occurs, will yield very good speedups. The parallel swapper can then be summarized as follows:

- **WHILE**: There are bad, untested elements in the heap:
  - Check the worst elements in parallel;
  - Reconnect if possible, and the neighbouring elements have not been reconnected (scalar, integer);
  - Remember swapped elements;
- **ENDWHILE**

#### 4.2 Removal of Bad Elements

A straightforward way to improve a mesh containing bad elements is to get rid of them. For tetrahedral grids this is particularly simple, as the removal of an internal edge does not lead to new element types for the surrounding elements. Once the bad elements have been identified (in parallel), they are compiled into a list and interrogated in turn. An element is removed by collapsing the points of one of the edges, as shown in Figure 7.

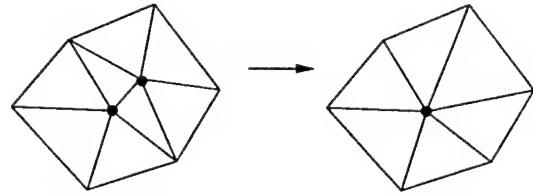


Figure 7 Removal of Element

This operation also removes all the elements that share this edge. It is advisable to make a check which of the points of the edge should be kept: point 1, point 2, or a point somewhere on the edge (e.g. the mid-point). This implies checking all elements that contain either point 1 or point 2. This procedure of removing bad elements is simple to implement and relatively fast. On the other hand, it will not tend to improve the overall quality of the mesh. It is therefore used mainly in a pre-smoothing or pre-optimization stage, where its main function is to eradicate elements of very bad quality from the mesh.

#### 4.3 Laplacian Smoothing

A number of smoothing techniques are lumped under this name. The edges of the triangulation are assumed to represent springs. These springs are relaxed in time using an explicit time stepping scheme, until an equilibrium of spring-forces has been established. Because 'globally' the variations of element size and shape are smooth, most of the non-equilibrium forces are local in nature. This implies that a significant improvement in mesh quality can be achieved rather

quickly. The force exerted by each spring is proportional to its length and along its direction. Therefore, the sum of the forces exerted by all springs surrounding a point can be written as:

$$\mathbf{f}_i = c \sum_{j=1}^{n_{s_i}} (\mathbf{x}_j - \mathbf{x}_i) ,$$

where  $c$  denotes the spring constant,  $\mathbf{x}_i$  the coordinates of the point, and the sum extends over all the points surrounding the point. The time-advancement for the coordinates is accomplished as follows:

$$\Delta \mathbf{x}_i = \Delta t \frac{1}{n_{s_i}} \mathbf{f}_i .$$

At the surface of the computational domain, no movement of points is allowed, i.e.  $\Delta \mathbf{x} = 0$ . Usually, the timestep (or relaxation parameter) is chosen as  $\Delta t = 0.8$ , and 5-6 timesteps yield an acceptable mesh. The parallelization of the Laplacian smoothing is similar to that of any other field solver. The loops over the elements/edges are colored so that cash misses are minimized, pipelining does not lead to memory contingencies and cache-line overwrite is avoided [Löh98b]. The application of the Laplacian smoothing technique will yield an overall improvement of grid quality, but can result in inverted or negative elements. These negative elements are eliminated. It has been found advisable to remove not only the negative elements, but also all elements that share points with them. This element removal gives rise to voids or holes in the mesh, which are regressed using the advancing front technique.

## 5. IMPLEMENTATION

The procedure described above was implemented using the shared memory (i.e. `c$doacross`) paradigm on the SGI Origin 2000. Although this is a distributed-memory machine, it can be programmed as a shared-memory machine. This choice was adopted for the following reasons:

- Coding for a shared-memory environment is much simpler than for a distributed-memory environment. The operating system takes care of most of the inherent message passing, communication conflicts, etc., relieving the user from this task;
- The production codes used in conjunction with the grid generator scale well using the shared-memory paradigm [Löh98b]. Even on 32 processors, more than 50% of the theoretical speedup is achieved. Figure 8 shows the speedups obtained for a steady compressible flow problem using an edge-based, upwind solver on different computer platforms.

Note that the speeds obtained using

the shared and distributed memory paradigms are comparable.

- For the last years, all of the large-scale production runs carried out by the author and his colleagues [Löh98a, Löh98c, Bau98, Bau99] were performed on these machines, using the shared-memory paradigm;

- The SGI Origin 2000 has become the dominant platform within the High-Performance Computing sites in the US; at present, there are more SGI Origin 2000 processors than those of all other vendors combined; the expectation is that this trend will not change drastically in the foreseeable future.

While some of the reasons stated above have to do with particular circumstances, the basic ideas of the proposed algorithms are general, and may be coded within a distributed memory framework with explicit message passing.

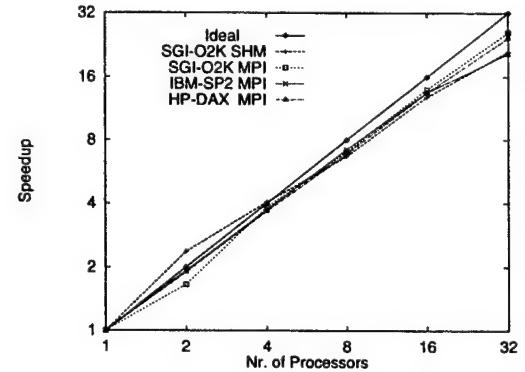


Figure 8 Performance of FEFLO on Different Platforms

## 6. EXAMPLES

The proposed parallel advancing front scheme has been used extensively over the last year in a production environment. We include a sampling of geometries gridded, highlighting the characteristics of the proposed scheme. The timings for all examples include surface gridding, mesh generation and mesh improvement, i.e. they can be considered representative of the timings obtained in a production environment. All timings were obtained on SGI Origin 2000 servers.

**6.1 Cube:** This academic example is included here to see how the procedure works in the 'best case scenario'. The unit cube is to be gridded with a uniform mesh of approximately 1 million tetrahedra. Although this is not a large grid, the timings shown in Figure 9 are illustrative.

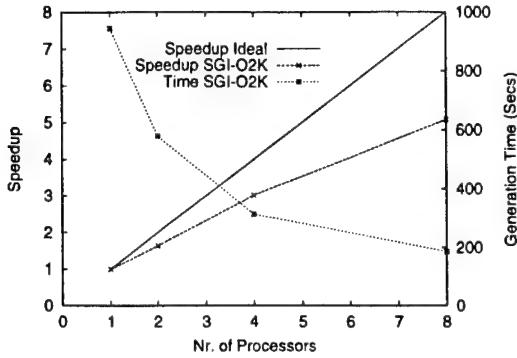


Figure 9 Cube: Speedups Obtained

**6.2 Aneurism:** This example is taken from a hemodynamic analysis recently conducted for this particular geometry. The outline of the domain, grid, as well as some sample results are shown in Figure 10.1.

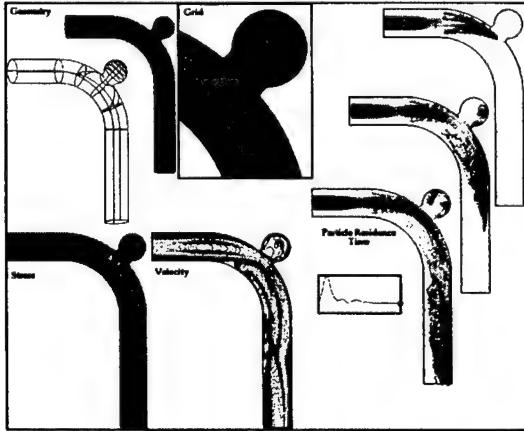


Figure 10.1 Aneurism

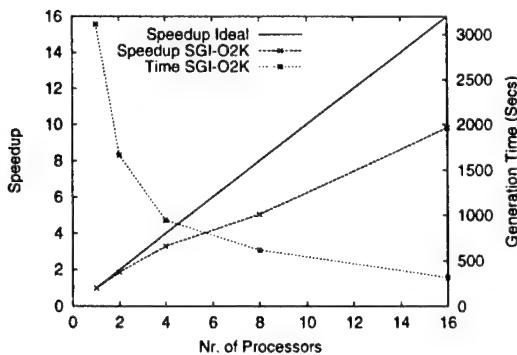


Figure 10.2 Aneurism: Speedups Obtained

The final uniform mesh had approximately 2.7 million tetrahedra. The speedup obtained is shown in Figure 10.2. As one can see, the parallel mesher was more efficient for this finer grid than for the unit cube, even though the volume to be gridded in each box can

vary widely due to geometric features.

**6.3 Garage:** This example was taken from a blast simulation recently carried out for an office complex. The outline of the domain is shown in Figure 11.1.

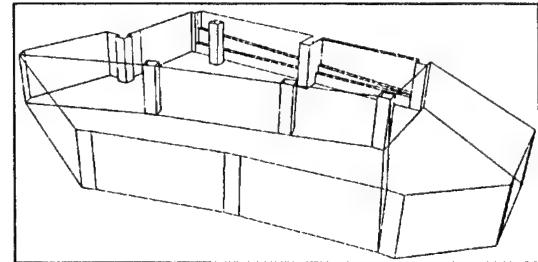


Figure 11.1 Garage: Wireframe

The final uniform mesh had approximately 9.2 million tetrahedra. The speedup obtained is shown in Figure 11.2. As before, the parallel mesher was more efficient for this finer grid than for the unit cube, even though the volume to be gridded in each box can vary widely due to geometric features.

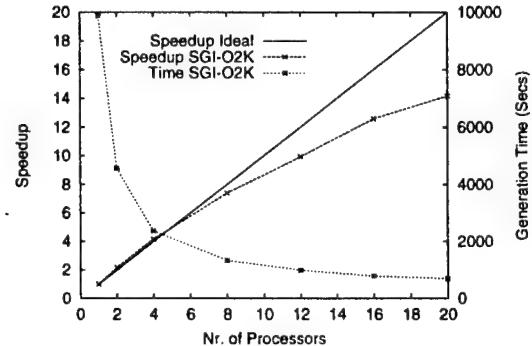


Figure 11.2 Garage: Speedups Obtained

**6.4 Tysons Corner:** This example was taken from a dispersion simulation recently carried out for this well-known shopping center. The outline of the domain is shown in Figures 12.1,12.2.

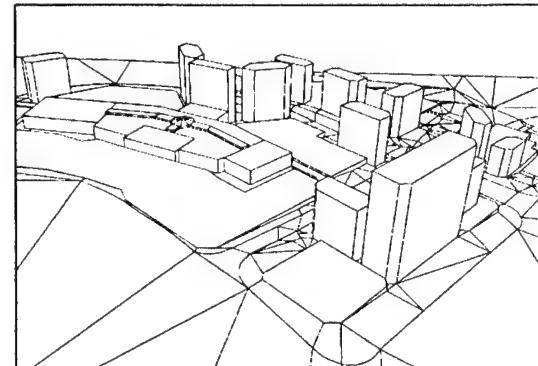


Figure 12.1 Tysons Corner: Wireframe (W)

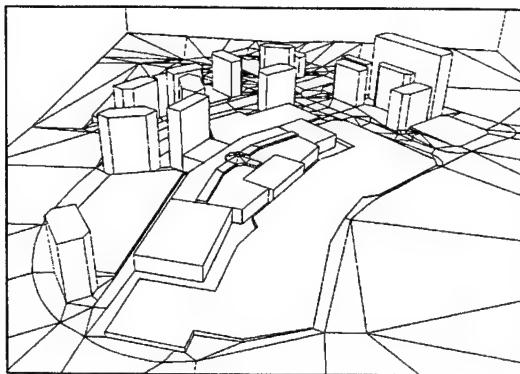


Figure 12.2 Tysons Corner: Wireframe (N)

The final mesh had approximately 16 million tetrahedra. The smallest and largest specified element side lengths were 230 cm and 1400 cm respectively. The speedup obtained is shown in Figure 12.3.

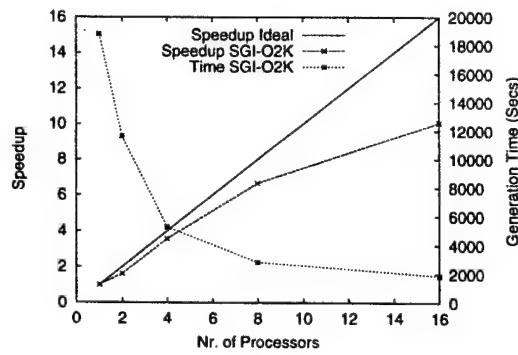


Figure 12.3 Tysons Corner: Speedups Obtained

**6.5 Space Shuttle:** This example is included here because it is typical of many aerodynamic data sets. The outline of the domain is shown in Figure 13.1.

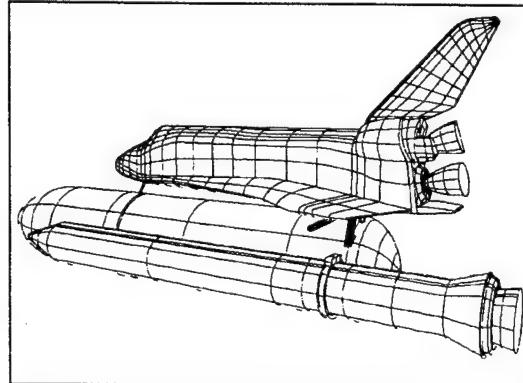


Figure 13.1 Space Shuttle: Outline of Domain

The surface triangulation of the final mesh, which had approximately 4 million tetrahedra, is shown in Figure 13.2. The smallest and largest specified element side lengths were 5.08 cm and 467.00 cm respectively,

i.e. an edge-length ratio of approximately 1 : 10<sup>2</sup> and a volume ratio of 1 : 10<sup>6</sup>. The spatial variation of element size was specified via approximately 200 sources [Löh96].

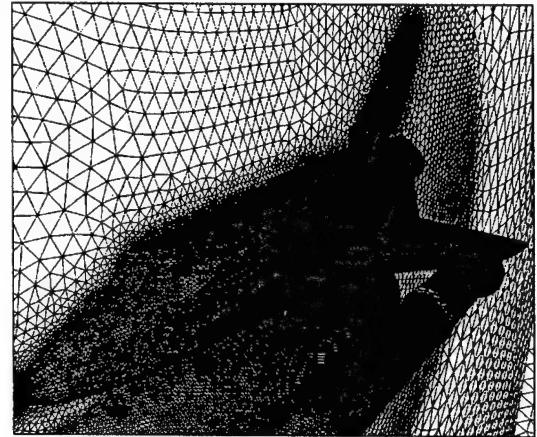


Figure 13.2 Space Shuttle: Surface Mesh

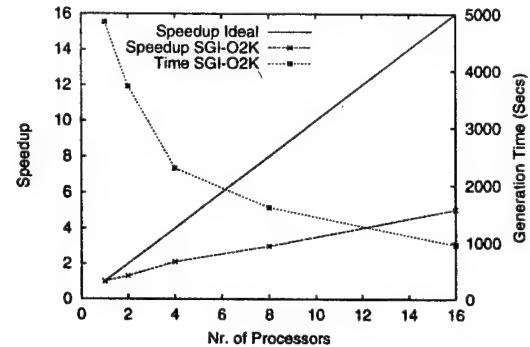


Figure 13.3 Space Shuttle: Speedups Obtained

The speedup obtained is shown in Figure 13.3. Finally, the results of an Euler run for an incoming Mach-nr. of  $M = 2.0$  and angle of attack of  $\alpha = 1.1^\circ$  are shown in Figure 13.4.

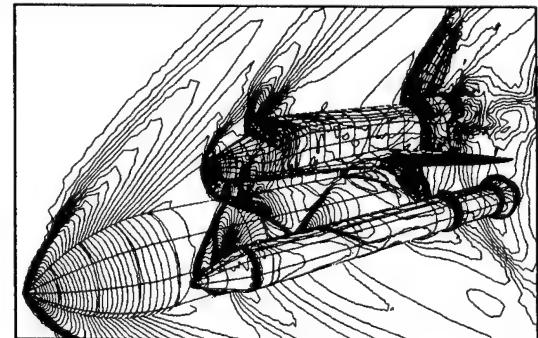


Figure 13.4 Space Shuttle: Surface Pressures

**6.6 Pilot Ejecting From F18:** Problems with moving bodies often require many remeshings during the course of a simulation. The case shown here was taken from a pilot ejection simulation recently conducted. The outline of the domain is shown in Figure 14.1. The surface triangulation of the final mesh, which had approximately 14 million tetrahedra, is shown in Figure 14.2. The smallest and largest specified element side lengths were 0.65 cm and 250.00 cm respectively, i.e. an edge-length ratio of approximately  $1 : 4 \cdot 10^2$  and a volume ratio of  $1 : 5.6 \cdot 10^7$ . The spatial variation of element size was specified via approximately 110 sources [Löh96].

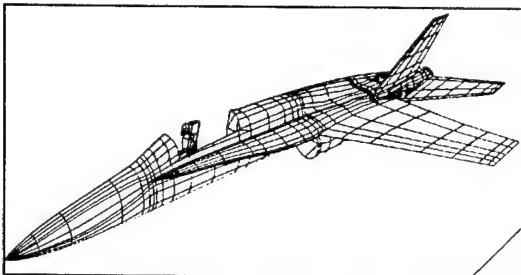


Figure 14.1 F18 Pilot Ejection: Outline of Domain

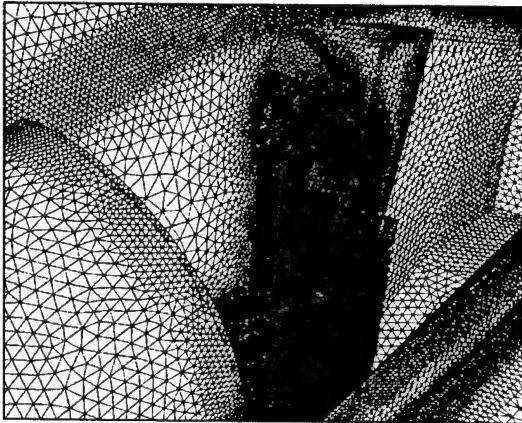


Figure 14.2 F18 Pilot Ejection: Surface Mesh (Closeup)

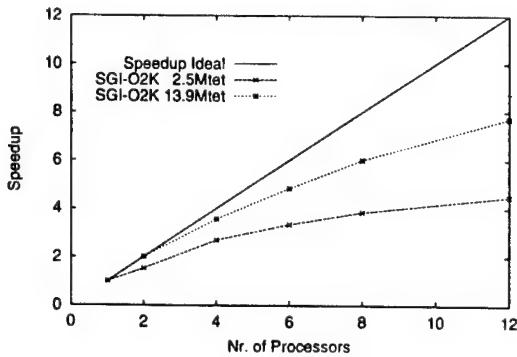


Figure 14.3 F18 Pilot Ejection: Speedups Obtained

The speedup obtained for two different grid sizes is displayed in Figure 14.3.

As could be seen from the previous examples, the grid generator certainly scales well with the number of processors. As with many other areas where parallel processing is being attempted, scalability improves with the amount of work required. The larger the grids, the better the scalability. One can also observe that for each case the 'scalability slope' is somewhat different, ranging from almost perfect for the garage to a 1:3 asymptote for the Shuttle. Closer inspection revealed that for the Shuttle, although the final mesh contains 4 million elements, the number of elements created in each parallel pass over increasing element sizes was rather modest, never exceeding 0.5 million elements. Thus, the inherent parallelism of this problem is rather low.

## 7. CONCLUSIONS AND OUTLOOK

A parallel advancing front scheme has been developed. The domain to be gridded is first subdivided spatially using a relatively coarse octree. Boxes are then identified and gridded in parallel. A scheme that resembles closely the advancing front technique on scalar machines is recovered by only considering the boxes of the active front that generate small elements. The procedure has been implemented on the SGI Origin class of machines using the shared memory paradigm. Timings for a variety of cases show speedups similar to those obtained for flow codes. The procedure has been used to generate grids for a large variety of cases, and is nearing production maturity. Current work is focusing on improved work prediction algorithms, as it is found that in going to a larger number of processors, any small imbalance incurs a heavy CPU penalty.

## 8. ACKNOWLEDGEMENTS

This research was partially supported by AFOSR, with Dr. Leonidas Sakell as the technical monitor.

## 9. REFERENCES

- [Bak89] T.J. Baker - Developments and Trends in Three-Dimensional Mesh Generation. *Appl. Num. Math.* **5**, 275-304 (1989).
- [Bau93] J.D. Baum, H. Luo and R. Löhner - Numerical Simulation of a Blast Inside a Boeing 747; *AIAA-93-3091* (1993).
- [Bau95] J.D. Baum, H. Luo and R. Löhner - Numerical Simulation of Blast in the World Trade Center; *AIAA-95-0085* (1995).
- [Bau96] J.D. Baum, H. Luo, R. Löhner, C. Yang, D. Pelessone and C. Charman - A Coupled

Fluid/Structure Modeling of Shock Interaction with a Truck; *AIAA-96-0795* (1996).

[Bau98] J.D. Baum, H. Luo and R. Löhner - The Numerical Simulation of Strongly Unsteady Flows With Hundreds of Moving Bodies; *AIAA-98-0788* (1998).

[Bau99] J.D. Baum, H. Luo, E. Mestreau, R. Löhner, D. Pelessone and C. Charman - A Coupled CFD/CSD Methodology for Modeling Weapon Detonation and Fragmentation; *AIAA-99-0794* (1999).

[Che97] L.P. Chew, N. Chrisochoides and F. Sukup - Parallel Constrained Delaunay Meshing; *Proc. 1997 Workshop on Trends in Unstructured Mesh Generation*, June (1997)

[dCo94] H.L. de Cougny, M.S. Shephard and C. Ozturan - Parallel Three-Dimensional Mesh Generation; *Computing Systems in Engineering* 5, 311-323 (1994).

[dCo95] H.L. de Cougny, M.S. Shephard and C. Ozturan - Parallel Three-Dimensional Mesh Generation on Distributed Memory MIMD Computers; *Tech. Rep. SCOREC Rep. # 7*, Rensselaer Polytechnic Institute (1995).

[Dar97] E. Darve and R. Löhner - Advanced Structured-Unstructured Solver for Electromagnetic Scattering from Multimaterial Objects; *AIAA-97-0863* (1997).

[Fry94] J. Frykestig - Advancing Front Mesh Generation Techniques with Application to the Finite Element Method; *Pub. 94:10*, Chalmers University of Technology; Göteborg, Sweden (1994).

[Geo91] P.L. George, F. Hecht and E. Saltel - Automatic Mesh Generator With Specified Boundary; *Comp. Meth. Appl. Mech. Eng.* 92, 269-288 (1991).

[Geo99] P.L. George - Tet Meshing: Construction, Optimization and Adaptation; *Proc. 8th Int. Meshing Roundtable*, South Lake Tahoe, October (1999).

[Has98] O. Hassan, L.B. Bayne, K. Morgan and N. P. Weatherill - An Adaptive Unstructured Mesh Method for Transient Flows Involving Moving Boundaries; pp. 662-674 in *Computational Fluid Dynamics '98* (K.D. Papailiou, D. Tsahalis, J. Périaux and D. Knörzer eds. ) Wiley (1998).

[Jin93] H. Jin and R.I. Tanner - Generation of Unstructured Tetrahedral Meshes by the Advancing Front Technique; *Int. J. Num. Meth. Eng.* 36, 1805-1823 (1993).

[Jou98] W. Jou - Comments on the Feasibility of LES for Commercial Airplane Wings; *AIAA-98-2801* (1998).

[Kam96] A. Kamoulakos, V. Chen, E. Mestreau and R. Löhner - Finite Element Modelling of Fluid/ Structure Interaction in Explosively Loaded Aircraft Fuselage Panels Using PAMSHOCK/ PAMFLOW Coupling; *Conf. on Spacecraft Structures, Materials and Mechanical Testing*, Noordwijk, The Netherlands, March (1996).

[Löh88a] R. Löhner - Some Useful Data Structures for the Generation of Unstructured Grids; *Comm. Appl. Num. Meth.* 4, 123-135 (1988).

[Löh88b] R. Löhner and P. Parikh - Three-Dimensional Grid Generation by the Advancing Front Method; *Int. J. Num. Meth. Fluids* 8, 1135-1149 (1988).

[Löh90] R. Löhner - Three-Dimensional Fluid-Structure Interaction Using a Finite Element Solver and Adaptive Remeshing; *Comp. Sys. in Eng.* 1, 2-4, 257-272 (1990).

[Löh92] R. Löhner, J. Camberos and M. Merriam - Parallel Unstructured Grid Generation; *Comp. Meth. Appl. Mech. Eng.* 95, 343-357 (1992).

[Löh96] R. Löhner - Progress in Grid Generation via the Advancing Front Technique; *Engineering with Computers* 12, 186-210 (1996).

[Löh98a] R. Löhner, C. Yang, J. Cebral, J.D. Baum, H. Luo, D. Pelessone and C. Charman - Fluid-Structure-Thermal Interaction Using a Loose Coupling Algorithm and Adaptive Unstructured Grids; *AIAA-98-2419* [Invited] (1998).

[Löh98b] R. Löhner - Renumbering Strategies for Unstructured- Grid Solvers Operating on Shared-Memory, Cache- Based Parallel Machines; *Comp. Meth. Appl. Mech. Eng.* 163, 95-109 (1998).

[Löh98c] R. Löhner, C. Yang and E. Oñate - Viscous Free Surface Hydrodynamics Using Unstructured Grids; *Proc. 22nd Symp. Naval Hydrodynamics*, Washington, D.C., August (1998).

[Mar95] D.L. Marcum and N.P. Weatherill - Unstructured Grid Generation Using Iterative Point Insertion and Local Reconnection; *AIAA J.* 33, 9, 1619-1625 (1995).

[Mav99] D.J. Mavriplis and S. Pirzadeh - Large-Scale Parallel Unstructured Mesh Computations for 3-D High-Lift Analysis; *ICASE Rep.* 99-9 (1999).

[Mes93] E. Mestreau, R. Löhner and S. Aita - TGV Tunnel-Entry Simulations Using a Finite Element Code with Automatic Remeshing; *AIAA-93-0890* (1993).

[Mes95] E. Mestreau and R. Löhner - Airbag Simulation Using Fluid/Structure Coupling; *AIAA-96-0798* (1996).

[Mor97] K. Morgan, P.J. Brookes, O. Hassan and N.P. Weatherill - Parallel Processing for the Simulation of Problems Involving Scattering of Electro-Magnetic Waves; in *Proc. Symp. Advances in Computational Mechanics* (L. Demkowicz and J.N. Reddy eds.) (1997).

[Oku96] T. Okusanya and J. Peraire - Parallel Unstructured Mesh Generation; *Proc. 5th Int. Conf. Num. Grid Generation in CFD and Related Fields*, Mississippi, April (1996).

[Oku97] T. Okusanya and J. Peraire - 3-D Parallel Unstructured Mesh Generation; *Proc. Joint ASME/ASCE/SES Summer Meeting* (1997).

[Per87] J. Peraire, M. Vahdati, K. Morgan and O.C. Zienkiewicz - Adaptive Remeshing for Compressible Flow Computations; *J. Comp. Phys.* 72, 449-466 (1987).

[Per88] J. Peraire, J. Peiro, L. Formaggia K. Morgan and O.C. Zienkiewicz - Finite Element Euler Calculations in Three Dimensions; *Int. J. Num. Meth. Eng.* 26, 2135-2159 (1988).

[Per90] J. Peraire, K. Morgan and J. Peiro - Unstructured Finite Element Mesh Generation and Adaptive Procedures for CFD; *AGARD-CP-464*, 18 (1990).

[Per92] J. Peraire, K. Morgan, and J. Peiro - Adaptive Remeshing in 3-D; *J. Comp. Phys.* (1992).

[Sai99] R. Said, N.P. Weatherill, K. Morgan and N.A. Verhoeven - Distributed Parallel Delaunay Mesh Generation; to appear in *Comp. Meth. Appl. Mech. Eng.* (1999).

[Sho95] A. Shostko and R. Löhner - Three-Dimensional Parallel Unstructured Grid Generation; *Int. J. Num. Meth. Eng.* 38, 905-925 (1995).

[Wea92] N.P. Weatherill - Delaunay Triangulation in Computational Fluid Dynamics; *Comp. Math. Appl.* 24, 5/6, 129-150 (1992).

[Wea94] N.P. Weatherill and O. Hassan - Efficient Three-Dimensional Delaunay Triangulation with Automatic Point Creation and Imposed Boundary Constraints; *Int. J. Num. Meth. Eng.* 37, 2005-2039 (1994).

[Yos98] S. Yoshimura, H. Nitta, G. Yagawa and H. Akiba - Parallel Automatic Mesh Generation Method of Ten-Million Nodes Problem Using Fuzzy Knowledge Processing and Computational Geometry; *Proc. 4th World CongComp. Mech.* Buenos Aires, Argentina, July (1998).

### APPENDIX 3: NAVIER-STOKES GRIDDING



**AIAA-99-0662**

**Generation of Unstructured Grids  
Suitable for RANS Calculations**

**Rainald Löhner**

**George Mason University, Fairfax, VA**

**37th AIAA Aerospace Sciences  
Meeting and Exhibit  
January 11-14, 1999 / Reno, NV**

## GENERATION OF UNSTRUCTURED GRIDS SUITABLE FOR RANS CALCULATIONS

Rainald Löhner

Institute for Computational Science and Informatics  
 M.S. 4C7, George Mason University  
 Fairfax, VA 22030-4444, USA

## ABSTRACT

A procedure for the generation of highly stretched grids suitable for Reynolds-Averaged Navier-Stokes (RANS) calculations is presented. In a first stage, an isotropic (Euler) mesh is generated. In a second stage, this grid is successively enriched with points in order to achieve highly stretched elements. The element reconnection is carried out using a constrained Delaunay approach. Points are introduced from the regions of lowest stretching towards the regions of highest stretching. The procedure has the advantages of not requiring any type of surface recovery, not requiring extra passes or work to mesh concave ridges/corners, and guarantees a final mesh, an essential requirement for industrial environments. Given that point placement and element quality are highly dependent for the Delaunay procedure, special procedures were developed in order to obtain optimal point placement.

## 1. INTRODUCTION

Many problems of Computational Mechanics are characterized by a very strong anisotropy in the spatial variation of the fields of interest. A typical example in fluid mechanics is a boundary layer. For laminar flow, the variations in the streamwise direction will be 3-5 orders of magnitude less than normal to it. The same applies to turbulent flows simulated using the Reynolds-Averaged Navier-Stokes (RANS) equations with suitable turbulence models. The reliable generation of high quality grids for RANS simulations has been attempted with varying degrees of success by several authors during the last decade (Nakahashi, 1987; Kallinderis, 1992; Löhner, 1993; Pirzadeh, 1994; Marcum, 1995; Pirzadeh, 1996; Peraire and Morgan, 1996-1998). Given that the generation of highly stretched grids is not trivial, and that computing power is increasing steadily, the immediate question that comes to mind is when RANS simulations will be replaced by Large-Eddy Simulations (LES) or even Direct Navier-Stokes (DNS) simulations. Let us assume an optimal mesh for LES simulations. This could be an adaptive Cartesian grid that consists of typical (large) Euler cells in the field and very small cells in the boundary layers in order to capture all relevant scales. Clearly, most points/cells will be located in the boundary layer regions. Denoting by  $N_p$  the number of points and by  $h$  the characteristic cell size, we have:

$$N_p \approx \frac{BL \text{Volume}}{h^3}$$

Copyright ©1999 by the author. Published by the AIAA with permission.

If we assume, conservatively, a laminar B-747 wing with a chord Reynolds-nr. of  $Re_m = 10^6$ , and furthermore assume that the boundary layer obeys the flat plate formula:

$$\frac{\delta_{lam}}{x} = \frac{5.5}{\sqrt{Re_x}}$$

the boundary layer thickness will be approximately  $\delta \approx 5 \cdot 10^{-3} \text{ m}$ , implying an (isotropic) element size of at most  $h \approx 5 \cdot 10^{-4} \text{ m}$ . The resulting number of gridpoints is then:

$$N_p = \frac{\delta \cdot A_{wing}}{h^2} = \frac{10 \cdot 250 \text{ m}^2}{(5 \cdot 10^{-4} \text{ m})^2} = 10^{10}$$

Current RANS production runs operate with  $N_p = 10^7$ . From Moore's Law (the doubling of computing power every 18 months), we can foresee LES grids in 15 years. As far as CPU is concerned, the number of timesteps  $N_t$  required to advect a particle accurately across the wing is proportional to the number of cells, i.e.

$$N_t = \frac{5m}{5 \cdot 10^{-4}m} = O(10^4)$$

Assuming the number of floating point operations per point per timestep to be  $N_{fpp} = O(10^3)$ , this would result in an operation count of

$$N_{ops} = O(10^{10}) \cdot O(10^4) \cdot O(10^3) = O(10^{17})$$

Given that the limit of human patience lies somewhere around  $O(10^3) \text{ sec}$ , the operation count obtained above implies a CPU performance requirement

of  $O(10^{14})$  FLOPS. Current production runs can operate at  $N_p = 10^{11}$  (100 GFLOPS). Invoking once more Moore's Law, we can foresee LES runs in 15 years. If we perform a sensitivity analysis, we note that the only linear component in these numbers was the human patience (e.g. 1hr to 1 day). As soon as we increase grid resolution by a factor of 10, we increase the number of points by  $10^3$ , the number of timesteps by 10, and the total effort by  $10^4$ , i.e. we have to wait yet another 20 years before we can carry out such a simulation. Faced with the pressing and immediate need to compute flows where Reynolds-number effects are important, we see that the optimal RANS gridding of geometrically complex domains is still an important topic of research.

## 2. THE RANS GRIDDING TECHNIQUE

The generation of isotropic unstructured grids has reached a fairly mature state, as evidenced by the many publications that have appeared over the last decade on this subject (Baker, 1987; Löhner and Parikh, 1988; Peraire et al., 1988; George et al., 1990; Joe, 1991; George, 1991; George and Hermeline, 1992; Löhner et al., 1992; Weatherill, 1992; Mavriplis, 1993; Jin and Tanner, 1993; Weatherill and Hassan, 1994; Marcum and Weatherill, 1995; Shostko and Löhner, 1995, Löhner, 1996) and the widespread use of unstructured grids in industry. The two most widely used techniques are the advancing front technique (Löhner and Parikh, 1988; Peraire et al., 1988; Löhner et al., 1992; Jin and Tanner, 1993; Shostko and Löhner, 1995, Löhner, 1996) and the Delaunay triangulation (Baker, 1987; Joe, 1991; George, 1991; George and Hermeline, 1992; Weatherill, 1992; Weatherill and Hassan, 1994; Marcum and Weatherill, 1995). Hybrid schemes, that combine an advancing front point placement with the Delaunay reconnection have also been used successfully (Merriam, 1991; Mavriplis, 1993; Müller, 1993). These isotropic mesh generation techniques have been used to generate grids with mildly stretched elements within the context of adaptive remeshing (Peraire, 1987; Löhner, 1988; Tilch, 1991; Habashi, 1998). However, they fail when attempting to generate highly stretched elements, a key requirement for Reynolds-Averaged Navier Stokes (RANS) calculations with turbulence models that reach into the sublayer.

A number of specialized schemes have been proposed to remedy this situation (Nakahashi, 1987; Kallinderis, 1992; Löhner, 1993; Pirzadeh, 1994; Marcum, 1995; Pirzadeh, 1996; Peraire and Morgan, 1996). The domain to be gridded was divided into isotropic and stretched element regions. In addition, a blending procedure to transition smoothly between these zones was provided. Typically, the stretched mesh region was generated first

(Kallinderis, 1992; Löhner, 1993; Pirzadeh, 1994; Marcum, 1995; Pirzadeh, 1996). Although we have used such a scheme (Löhner, 1993) for a number of years, we have found several situations in which the requirement of a semi-structured element or point placement close to wetted surfaces is impossible, prompting us to search for a more general technique. This procedure may be summarized as follows:

- Generate an isotropic mesh; this can be done with any unstructured grid generator;
- Remove all points in regions where stretched elements are to be generated;
- Using a constrained Delaunay technique, introduce points in order to generate highly stretched elements;
- Introduce the points in ascending levels of stretching, i.e. from the domain interior to the boundary.

This procedure has the following advantages:

- No surface recovery is required for the Delaunay reconnection, eliminating the most problematic part of this technique;
- Proper meshing of concave ridges/corners is obtained;
- The meshing of concave ridges/corners requires no extra work; this important advantage is shown in Figure 1;
- Meshing problems due to surface curvature are minimized;
- In principle, no CAD representation of the surface is required; and
- A final mesh is guaranteed, an essential requirement for automation.

The disadvantages are the following:

- As with any Delaunay technique, the mesh quality is extremely sensitive to point placement.

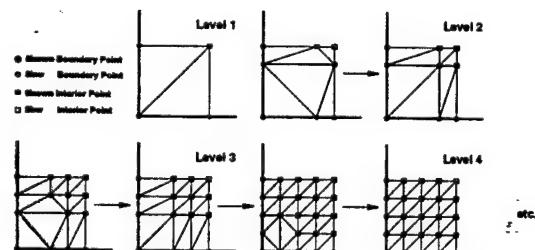


Figure 1 Introduction of Points at Corners

## 3. INSERTION OF POINTS

The insertion of points is carried out using the constrained Delaunay procedure (George, 1991), which may be summarized as follows. Given a new point  $i$  at location  $\mathbf{x}_i$ :

- Find the element(s)  $\mathbf{x}_i$  falls into;

- Obtain all elements whose circumsphere encompasses  $x_i$ ;
- Remove from this list of elements all those that would not form a proper element (volume, angles) with  $x_i$ ; this results in a properly constrained convex hull;
- Reconnect the outer faces of the convex hull with  $x_i$  to form new elements.

The procedure has been sketched in Figure 2.

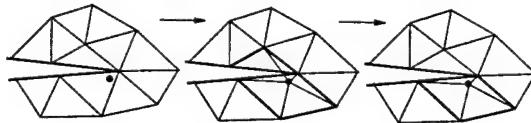


Figure 2 Introduction of Field Points

For boundary points some additional steps are required. Given a new point boundary point  $i$  at location  $x_i$ :

- Determine if the point is on a boundary edge or face;
- Reconnect these elements without regard to the Delaunay criterion;
- Find the element(s)  $x_i$  falls into;
- Obtain all elements whose circumsphere encompasses  $x_i$ ;
- Remove from this list of elements all those that would not form a proper element (volume, angles) with  $x_i$ ; this results in a properly constrained convex hull;
- Reconnect the boundary faces (see Figure 3);
- Reconnect the outer faces of the convex hull with  $x_i$  to form new elements.

The reconnection of boundary faces is carried out by diagonal swapping. For curved surfaces, it is necessary to apply angle constraints in order not to lose surface resolution/definition or surface patch integrity.

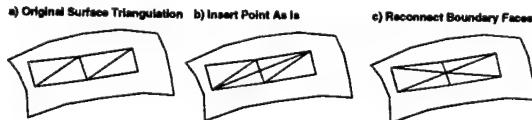


Figure 3 Introduction of Surface Points

The points are inserted according to layers following the normals emanating from 'wetted' surface points, starting from the outermost layer, and moving towards the boundaries or wake centerlines. Points are only introduced if the spacing normal to the wall is below a fraction of the isotropic element size specified by the user at the particular location. This is

important for grids with a large variation of element size, and produces a smooth transition from the Euler region into the RANS region.

#### 4. ADDITION OF EXTRA SURFACE POINTS

For complex geometries with narrow surface strips close to concave edges, it is not possible to obtain a good surface mesh unless one introduces further points in these regions. A typical situation is shown in Figure 4.

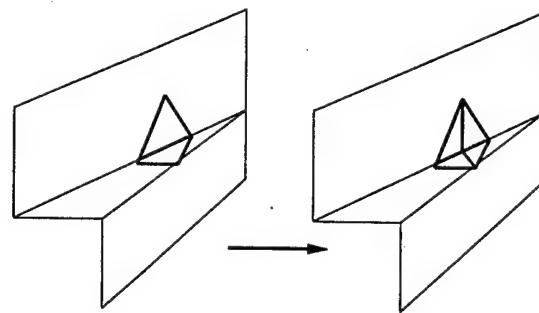


Figure 4 Addition of Points Along a Ridge

These additional points are introduced by identifying the corners where potential problems can appear. These are typically concave, and are characterized by normals that would introduce points close to another edge. Subsequent reconnection using the constrained Delaunay technique would yield elements with very large angles. In order to avoid this, additional points are introduced along the concave edge to mitigate this topological effect.

#### 5. CONSTRUCTION OF NORMALS

The insertion of points to construct highly stretched elements is carried out along normals that may start either on the boundary (boundary layers) or in the field (wakes). The number of normals emanating from a surface point can vary, depending on whether we have a convex or concave surface. Figure 5 shows just a few of a large class of cases that have to be considered.

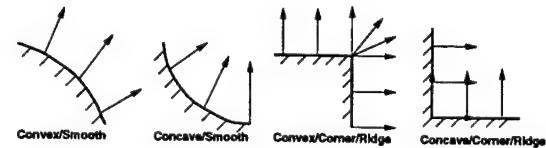


Figure 5 Some Possible Cases for Surface Normals

## 6. REMOVAL OF POINTS BEFORE RANS GRIDDING

Since the quality of grids generated using the Delaunay technique is very sensitive to point placement, it is advisable to remove any (isotropic) points that may interfere with the semi-structured points in the highly stretched regions. The regions where this could happen can be obtained before starting the RANS meshing. The point removal algorithm may be summarized as follows:

- For each point marked for removal:
  - Obtain all edges touching this point, and the corresponding neighbouring points;
  - Remove the edges that can not be collapsed due to topological considerations (end-, line-, surface- or volume points);
  - Remove the edges that, if collapsed, would lead to small or negative elements;
  - Remove the edges that, if collapsed, would lead to very small or very large angles;
  - Order the remaining edges according to their length;
  - Remove the marked edges, renumbering the elements;
- Compress and renumber the element and point lists.

The effectiveness of this point removal algorithm may be enhanced by combining it with edge and face swapping and point movement. The final point removal algorithm then takes the form of the following loop:

- DO: For a maximum number of passes:
  - Remove marked points by edge-collapse;
  - IF: Points could be removed:
    - Perform edge and face swapping;
  - ELSE
    - Move points that could not be removed;
    - Perform edge and face swapping;
  - ENDIF
  - ENDDO

The advantage of moving points can be seen from the small example shown in Figure 6. Any edge collapse for the point in the center of the domain would lead to elements with vanishing volumes. Mesh movement in combination with diagonal swapping removes this quandary. This combination of point movement and edge swapping is very efficient, typically leaving only 0.1% of the points marked for removal in the mesh.

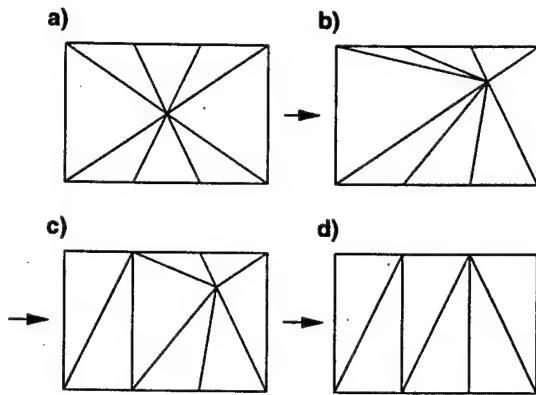


Figure 6 Movement and Removal of Points

## 7. POINT INTRODUCTION FOR GAPS

In narrow gaps, the introduction of points from two close surfaces covered with RANS grids can lead to very poor mesh quality. Figure 7 shows an example where the resulting mesh is clearly inappropriate for RANS calculations. In order to avoid the introduction of points in such regions, the host element into which the new point falls is checked for proximity to another surface. If any of the points of this element are on the surface and are too close to the new point, the new point is rejected. We remark that the Delaunay reconnection procedure requires the determination of the host element, so that this check incurs only a modest amount of CPU.

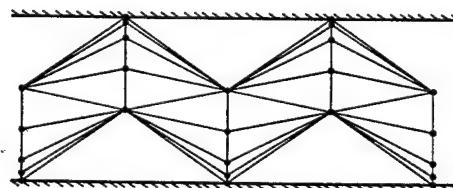


Figure 7 Gridding of Gaps

## 8. EXAMPLES

The described RANS gridding procedure has been operational for approximately a year, and was used to mesh a series of test cases. Four of these are included here.

- a) Nose-Cone: The surface mesh of the isotropic mesh is shown in Figure 8.1. The removal of points within the future non-isotropic layers of elements results in the surface mesh shown in Figure 8.2. The final surface mesh is depicted in Figure 8.3. One can see the considerable stretching achieved.

b) Ship: This case shows a RANS grid for the generic chemical tanker shown in Figure 9.1. Figures 9.2 and 10.1-10.3 show surface grids and cross-sections for the generated mesh. The isotropic (Euler) mesh for this case had approximately 1.2 million elements, while the final, anisotropic (RANS) mesh had close to 5 million elements.

c) Flyer: The third configuration considered is that of a generic hypersonic flyer. Figures 11.1-11.2 show the surface grids obtained, and Figures 12.1-12.2 show a plane cut in the region of the vertical tail and stabilizer. Observe the proper gridding in the corner regions. The isotropic (Euler) mesh for this case had approximately 2 million elements, while the final, anisotropic (RANS) mesh had approximately 6 million elements. On an SGI Origin 2000, using 1 R10000 processor, the isotropic grid generation took approximately 45 minutes, while the anisotropic enrichment took approximately 40 minutes. One can see from these figures that the speed of the proposed RANS gridding technique is acceptable.

d) Racecar: The fourth configuration considered is that of a generic racecar. Figures 13.1-13.2 show the surface definition and the overall surface grid, while Figures 14-15 focus on particular regions of the mesh (driver, nose of the car, back of the car, gaps, etc.), which had approximately 8.3 million elements. A result for a flow simulation using Luo's implicit LU-SGS-GMRES solver (Luo 1998) is shown in Figures 16.1-16.2, demonstrating the usefulness of the procedure.

## 9. CONCLUSIONS AND OUTLOOK

A procedure for the generation of highly stretched grids suitable for Reynolds-Averaged Navier-Stokes (RANS) calculations has been developed. In a first stage, an isotropic (Euler) mesh is generated. In a second stage, this grid is successively enriched with points in order to achieve highly stretched elements. The element reconnection is carried out using a constrained Delaunay approach. Points are introduced from the regions of lowest stretching towards the regions of highest stretching. The procedure has the advantages of not requiring any type of surface recovery, not requiring extra passes or work to mesh concave ridges/corners, and guarantees a final mesh, an essential requirement for industrial environments. Given that point placement and element quality are highly dependent for the Delaunay procedure, special procedures are required in order to obtain optimal point placement. Among these, the most important is the removal of points that fall into the RANS zone before enriching the mesh.

Several examples demonstrate the usefulness of the proposed anisotropic gridding technique. Timings

from grids generated to date show that the procedure is faster than traditional advancing front techniques for isotropic grids, implying that the generation of anisotropic grids does not place an extra burden on CPU or memory requirements.

Near-future work will center on:

- Improvements in point placement and element control;
- Automatic wake placement and topological connection to CAD models; and
- Parallelization.

Looking further into the future, we envision fully automatic RANS gridders integrated into a multidisciplinary, database-linked framework that is accessible anywhere on demand, simulations with unprecedented detail and realism carried out in fast succession, and first-principles driven virtual reality.

## 10. ACKNOWLEDGEMENTS

The author gratefully acknowledges the many insightful discussions with Prof. P.L. George (INRIA, France), which were instrumental in achieving a fast and robust Delaunay triangulation tool. The data for the generic racecar was provided by Prof. J. Katz (SDSU), and the actual flow solution was obtained by Dr. H. Luo (SAIC).

This work was partially supported by AFOSR, with Dr. Leonidas Sakell as the technical monitor.

## 11. REFERENCES

Baker, T.J. - Three-Dimensional Mesh Generation by Triangulation of Arbitrary Point Sets; *AIAA-CP-87-1124*, 8th CFD Conf., Hawaii (1987).

George, P.L., F. Hecht and E. Saltel - Fully Automatic Mesh Generation for 3D Domains of any Shape; *Impact of Computing in Science and Engineering* 2, 3, 187-218 (1990).

George, P.L. - *Automatic Mesh Generation*; J. Wiley & Sons (1991).

George, P.L., F. Hecht and E. Saltel - Automatic Mesh Generator With Specified Boundary; *Comp. Meth. Appl. Mech. Eng.* 92, 269-288 (1991).

George P.L., and F. Hermeline - Delaunay's Mesh of a Convex Polyhedron in Dimension D. Application to Arbitrary Polyhedra; *Int. J. Num. Meth. Eng.* 33, 975-995 (1992).

Habashi, W.G., M Fortin, J. Dom pierre, M.-G. Vallet and Y. Bourgault - Anisotropic Mesh Adaptation: A Step Towards A Mesh-Independent and User-Independent CFD; pp. 99-117 in *Barriers and Challenges in CFD* (Venkatakrishnan et al. eds.), Kluwer (1998).

Jin, H. and R.I. Tanner - Generation of Unstructured Tetrahedral Meshes by the Advancing Front Technique; *Int. J. Num. Meth. Eng.* 36, 1805-1823 (1993).

Joe, B. - Construction of Three-Dimensional Delaunay Triangulations Using Local Transformations; *Computer Aided Geometric Design* 8, 123-142 (1991).

Joe, B. - Delaunay Versus Max-Min Solid Angle Triangulations for Three-Dimensional Mesh Generation; *Int. J. Num. Meth. Eng.* 31, 987-997 (1991).

Kallinderis, Y. and S. Ward - Prismatic Grid Generation with an Efficient Algebraic Method for Aircraft Configurations; *AIAA-92-2721* (1992).

Löhner, R. and P. Parikh - Three-Dimensional Grid Generation by the Advancing Front Method; *Int. J. Num. Meth. Fluids* 8, 1135-1149 (1988).

Löhner, R. - An Adaptive Finite Element Solver for Transient Problems with Moving Bodies; *Comp. Struct.* 30, 303-317 (1988).

Löhner, R., J. Camberos and M. Merriam - Parallel Unstructured Grid Generation; *Comp. Meth. Appl. Mech. Eng.* 95, 343-357 (1992).

Löhner, R. - Matching Semi-Structured and Unstructured Grids for Navier-Stokes Calculations; *AIAA-93-3348-CP* (1993).

Löhner, R. - Progress in Grid Generation via the Advancing Front Technique; *Engineering with Computers* 12, 186-210 (1996).

Luo, H., J.D. Baum and R. Löhner - A Fast, Matrix-Free Implicit Method for Compressible Flows on Unstructured Grids; *J. Comp. Phys.* 146, 664-690 (1998).

Marcum, D.L. and N.P. Weatherill - Unstructured Grid Generation Using Iterative Point Insertion and Local Reconnection; *AIAA J.* 33, 9, 1619-1625 (1995).

Marcum, D.L. - Generation of Unstructured Grids for Viscous Flow Applications; *AIAA-95-0212* (1995).

Mavriplis, D.J. - An Advancing Front Delaunay Triangulation Algorithm Designed for Robustness; *AIAA-93-0671* (1993).

Merriam, M. - An Efficient Advancing Front Algorithm for Delaunay Triangulation; *AIAA-91-0792* (1991).

Müller, J., P.L. Roe and H. Deconinck - A Frontal Approach for Internal Node Generation in Delaunay Triangulations; *Int. J. Num. Meth. Eng.* 17, 2, 241-256 (1993).

Nakahashi, K. - FDM-FEM Zonal Approach for Viscous Flow Computations over Multiple Bodies; *AIAA-87-0604* (1987).

Peraire, P. M. Vahdati, K. Morgan and O.C. Zienkiewicz - Adaptive Remeshing for Compressible Flow Computations; *J. Comp. Phys.* 72, 449-466 (1987).

Peraire, J., J. Peiro, L. Formaggia, K. Morgan and O.C. Zienkiewicz - Finite Element Euler Calculations in Three Dimensions; *Int. J. Num. Meth. Eng.* 26, 2135-2159 (1988).

Peraire, J. and K. Morgan - Unstructured Mesh Generation Including Directional Refinement for Aerodynamic Flow Simulation; *Proc. 5th Int. Conf. Num. Grid Generation in CFD and Related Fields*, Mississippi, April (1996).

Peraire, J. and K. Morgan - Unstructured Mesh Generation Including Directional Refinement for Aerodynamic Flow Simulation; *AIAA-98-3010* (1998).

Pirzadeh, S. - Viscous Unstructured Three-Dimensional Grids by the Advancing-Layers Method; *AIAA-94-0417* (1994).

Pirzadeh, S. - Progress Towards a User-Oriented Unstructured Viscous Grid Generator; *AIAA-96-0031* (1996).

Shostko, A. and R. Löhner - Three-Dimensional Parallel Unstructured Grid Generation; *Int. J. Num. Meth. Eng.* 38, 905-925 (1995).

Tilch, R. - PhD Thesis, CERFACS, Toulouse, France (1991).

Weatherill, N.P. - Delaunay Triangulation in Computational Fluid Dynamics; *Comp. Math. Appl.* 24, 5/6, 129-150 (1992).

Weatherill, N.P. and O. Hassan - Efficient Three-Dimensional Delaunay Triangulation with Automatic Point Creation and Imposed Boundary Constraints; *Int. J. Num. Meth. Eng.* 37, 2005-2039 (1994).

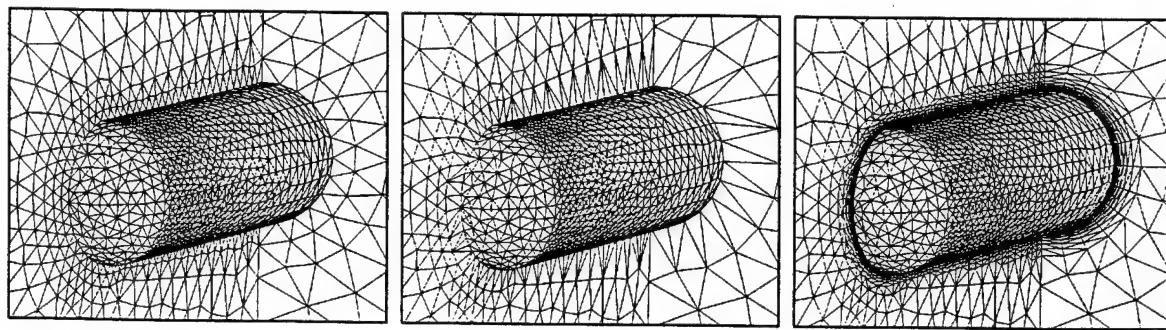


Figure 8 Surface of: Isotropic Mesh, After Element Removal and Final Mesh

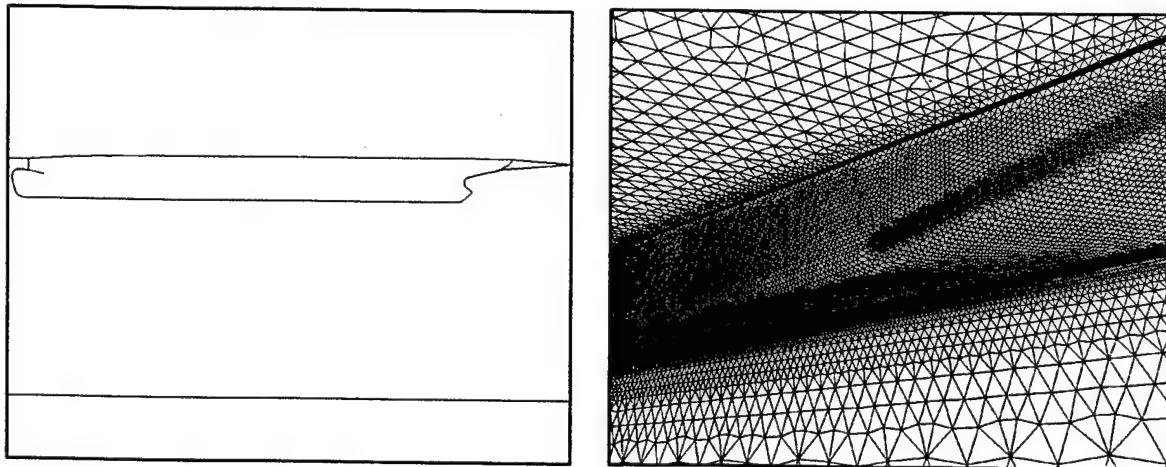


Figure 9 Computational Domain and Surface Mesh Along the Side

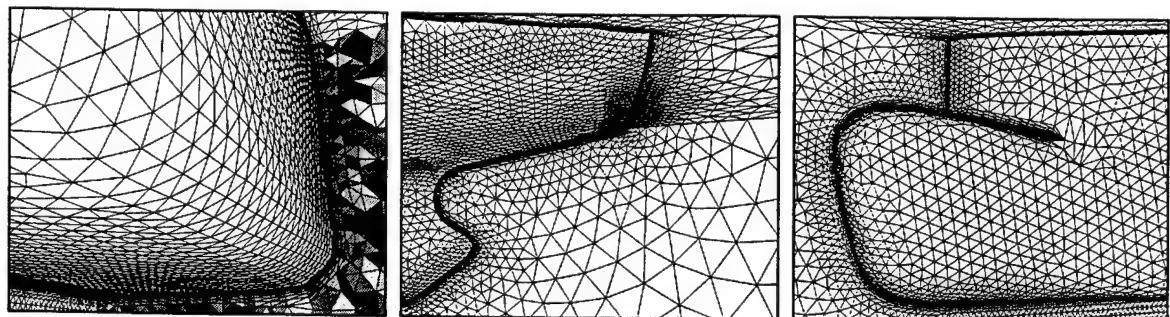


Figure 10 Transversal Cut, Stern Region and Bow Region

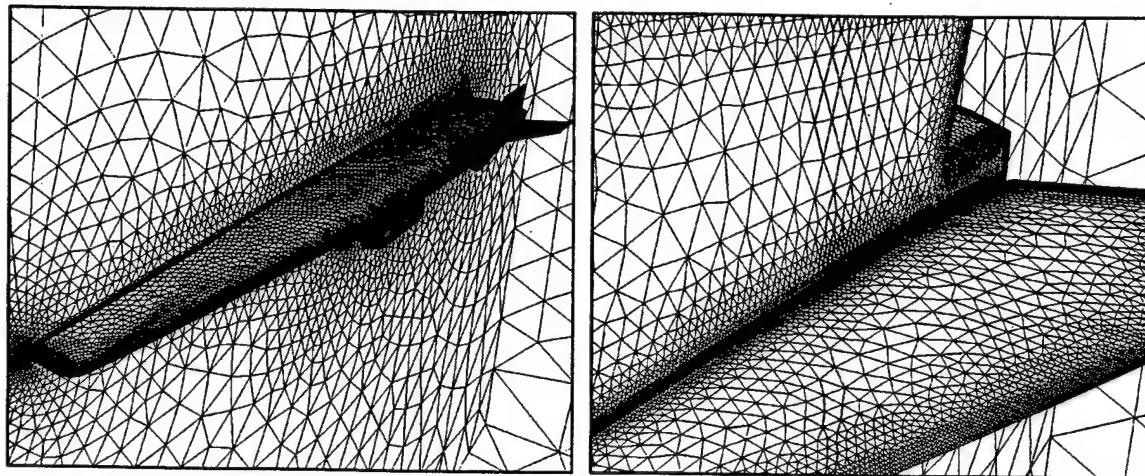


Figure 11 Generic Hypersonic Flyer: Surface Grid and Detail

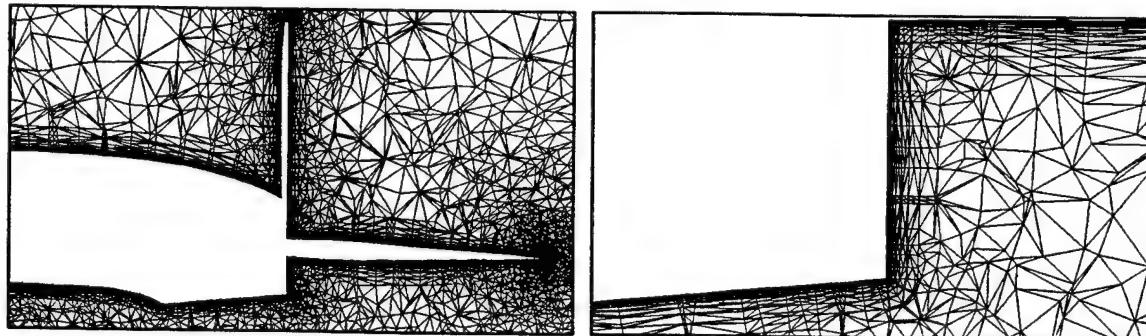


Figure 12 Generic Hypersonic Flyer: Planar Cut and Detail

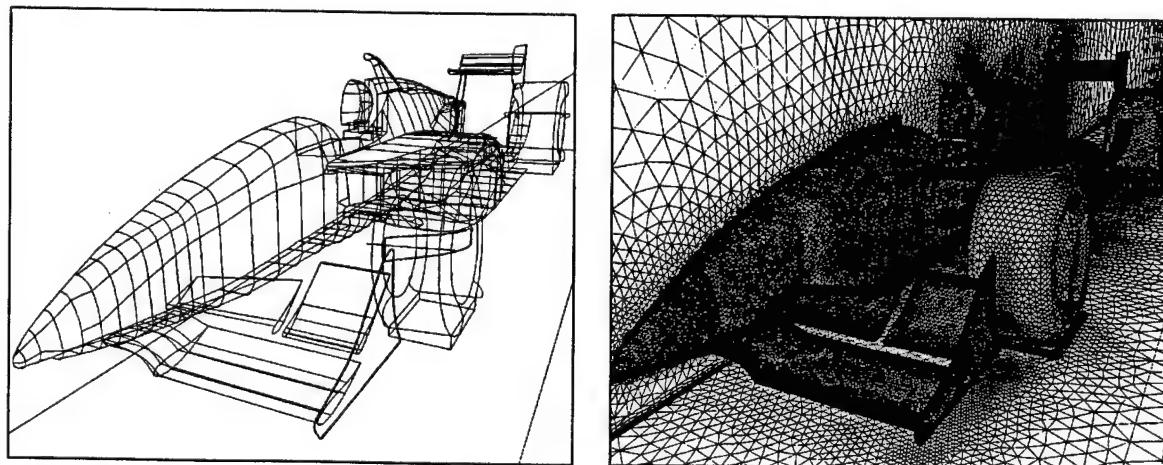


Figure 13 Generic Racecar: Surface Definition and Surface Mesh

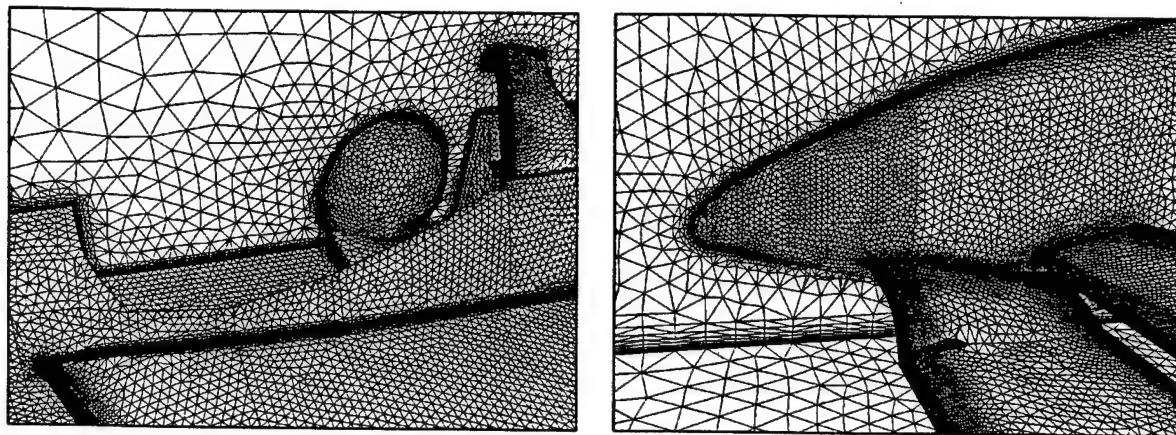


Figure 14 Driver and Nose of the Car

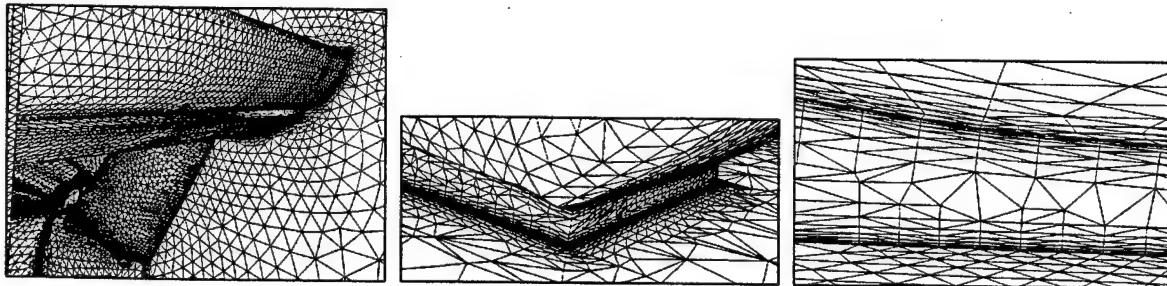


Figure 15 Details of Back, Back-Wheel and Car-Floor Gap

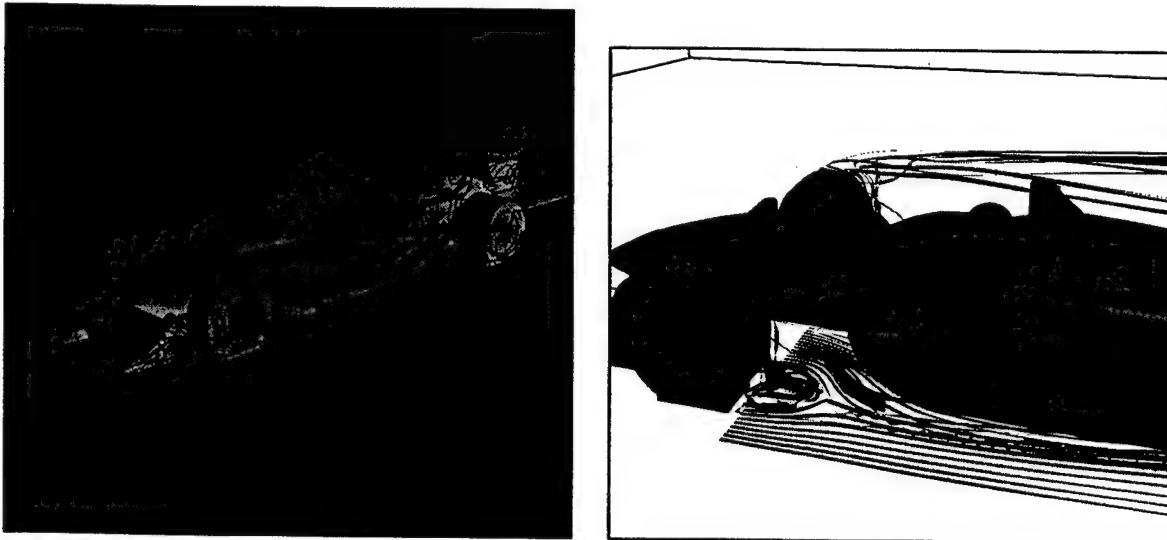


Figure 16 Pressure Contours and Streamlines

#### APPENDIX 4: FLUID-STRUCTURE-THERMAL INTERACTION



**AIAA-98-2419**

**Fluid-Structure-Thermal Interaction Using  
A Loose Coupling Algorithm  
And Adaptive Unstructured Grids**

**Rainald Löhner<sup>1</sup>, Chi Yang<sup>1</sup>, Juan Cebral<sup>1</sup>,  
Joseph D. Baum<sup>2</sup>, Hong Luo<sup>2</sup>,  
Daniele Pelessone<sup>3</sup> and Charles Charman<sup>3</sup>**

<sup>1</sup>George Mason University, Fairfax, VA

<sup>2</sup>SAIC, McLean, VA

<sup>3</sup>General Atomics, San Diego, CA

**29th AIAA Fluid Dynamics Conference  
June 15-18, 1998 / Albuquerque, NM**

**FLUID-STRUCTURE-THERMAL INTERACTION USING  
A LOOSE COUPLING ALGORITHM  
AND ADAPTIVE UNSTRUCTURED GRIDS**

Rainald Löhner<sup>1</sup>, Chi Yang<sup>1</sup>, Juan Cebral<sup>1</sup>,  
Joseph D. Baum<sup>2</sup>, Hong Luo<sup>2</sup>,  
Daniele Pelessone<sup>3</sup> and Charles Charman<sup>3</sup>

<sup>1</sup>GMU/CSI, George Mason University, Fairfax, VA 22030, USA

<sup>2</sup>Science Applications International Corporation  
1710 Goodridge Drive, MS 2-3-1, McLean, VA 22102, USA

<sup>3</sup>General Atomics, San Diego, CA 92121, USA

**ABSTRACT**

We present a loosely coupled algorithm to combine Computational Fluid Dynamics (CFD), Computational Structural Dynamics (CSD) and Computational Thermo-Dynamics (CTD) codes in order to solve, in a cost-effective manner, fluid-structure-thermal interaction problems. The basic fluid-, structural- and thermodynamics codes are altered as little as possible. The structure is used as the 'master-surface' to define the extent of the fluid region, and the fluid is used as the 'master-surface' to define the mechanical and thermal loads. The transfer of loads, heat fluxes, displacements, velocities and temperatures is carried out via fast interpolation and projection algorithms. As shown, this fluid-structure-thermal algorithm can be interpreted as an iterative solution to the fully-coupled, large matrix problem that results from the discretization of the complete problem. Results from steady supersonic flow and shock-structure interaction problems indicate that the proposed approach offers a convenient and cost-effective way of coupling CFD, CSD and CTD codes without a complete re-write of them.

**1. INTRODUCTION**

The advent of supercomputing over the last decade has led to fairly mature codes in all core disciplines of Computational Physics. Due to their immediate importance to industry, a large number of public-domain and commercial codes exists for the simulation of Computational Fluid Dynamics (CFD), Computational Structural Dynamics (CSD) and Computational Thermo-Dynamics (CTD) [Cod98]. There exist large classes of important engineering problems that require the concurrent application of CFD, CSD and CTD techniques. Some examples are:

- Explosive inflation, e.g. for airbags, where flow-field, fabric loading and fabric temperature are closely linked;
- Hypersonic Flight, where the deformation of the structure due to aerodynamic and aerothermal loads is such that a significant variation of the flowfield takes place (shock location, surface heating, etc.), and
- Process modeling, where the heat due to shear and chemical reactions in the fluid gives rise to non-uniform temperature distributions in the

solid, resulting in thermal stress loads and the associated fatigue.

- Variable Geometry Vehicles, where the change of geometry implies a transient phase in which structures and flowfields are interacting strongly, and, in most cases, non-linearly.

Most of these problems are presently solved either iteratively, i.e. making several cycles of 'CFD run followed by CTD run followed by CSD run', or by assuming that the CFD, CSD and CTD problems can be decoupled 'to first order'. Throughout most large manufacturing companies the respective CFD, CSD and CTD runs are performed in different divisions which may be geographically dispersed, leading to time-delays, loss of information, and, most importantly, loss of insight.

The need to solve fluid-structure-thermal interaction problems has prompted a number of developments in this field in recent years. The best way to sort these efforts is by classifying them according to the physical and numerical complexity employed for the fluid and structure respectively (see Figure 1). For the fluid, the PDEs solved are, in increasing order of physical complexity:

- F1. Laplace/Helmholtz Operators (inviscid, irrotational, isentropic flow),

- F2. Non-Linear Laplace Operators (inviscid, irrotational flow),
- F3. Euler Equations (inviscid flow),
- F4. Reynolds-Averaged Navier-Stokes Equations (viscous, time-averaged flow),
- F5. Large-Eddy Simulations (viscous flow with spatio-temporal cut-off), and
- F6. Navier-Stokes Equations.

Each of these approximations requires between one and two orders of magnitude more CPU-time and memory than the preceding one. For the linear case, boundary element methods may be employed, whereas all other approximations are typically approximated on a grid with spatial discretizations obtained from Finite Difference, Finite Volume, Finite Element, or Spectral Element techniques.

For the structure, the PDEs solved are, in increasing order of physical complexity:

- S1. 6 Degrees of Freedom Integration (rigid body),
- S2. Linear Elastic Models, either through
  - a) A Modal Decomposition, or
  - b) A Finite Element Discretization,
- S3. Elasto-Plastic Models, and
- S4. Elasto-Plastic Models with Contact, Rupture, etc.

As before, each of these approximations requires between one and two orders of magnitude more CPU-time and memory than the preceding one. For structures, the spatial discretization is typically carried out using Finite Element techniques [Zie88].

For the thermodynamic description of the problem, the PDEs solved are, in increasing order of physical complexity:

- T1. Adiabatic or Isothermal Walls and Sources/Sinks,
- T2. Linear Heat Conduction, either through
  - a) A Network Decomposition [Gas87] or
  - b) A Finite Element Discretization [Zie88, Pro92, Löh94], and
- T3. Nonlinear Heat Conduction [Löh94].

As before, each of these approximations requires between one and two orders of magnitude more CPU-time and memory than the preceding one. When solving the PDE's describing the heat flux, the spatial discretization is usually carried out using Finite Element techniques [Zie88].

A major characteristic of fluid-structure-thermal algorithms is the requirement to combine the discretizations for the fluid and the structure. This provides a fourth classification item (see Figure 2):

- I1. Same surface/volume discretization;
- I2. Different surface/volume discretization coupled via:
  - a) Interpolation,
  - b) Least-Squares,
  - c) Lagrange Multipliers, or

d) A Third, so-called 'Virtual' Surface Grid. For the simple CSD approximations S1,S2a, there is no discretization of the structure *per se*, so that the transfer of information between fluid and structure is straightforward.

With this series of possibilities, we are now in a position to classify previous fluid-structure-thermal interaction work. The three classic fields that combine two of these disciplines: structural acoustics and aeroelasticity for fluid-structure interaction and thermal stress analysis for structures have seen the largest amount of activity, particularly in those instances where the fluid, structure and heat transfer were assumed as linear (inviscid, irrotational, isentropic fluid, linear elastic structure, linear heat conduction). Of the many references in aeroelastics, we mention [Jac87, Bat88, Gur90, Eve90, Eve91, Bos93, Rau93, Fel93, Gur93, Alo95]. For thermoelastics, see [Tho88, Tam97]. To our knowledge, the earliest effort to simulate fully coupled fluid/ structure/ thermal problem is the work of Thornton [Tho88], which was carried out in 2-D.

The present effort is directed towards nonlinear applications, in particular structures that undergo deformations due to aerodynamic or aero-thermodynamic loads. For this reason, we start immediately with the Euler and Reynolds-Averaged Navier-Stokes equations for the fluid, and either the linear elastic (for aerospace flight vehicles) or nonlinear, large-deformation equations (for impact simulations) for the structure. Given that the geometrical complexity of the problems targeted for simulation can be severe and the deformation considerable, automatic grid generation is a prime requirement. For this reason, unstructured grids are employed for the fluid and the structures. The elements used for the fluid are tetrahedral, whereas the elements for the structure are typically trusses, beams, quads or bricks.

The remainder of the paper is organized as follows: Section 2 describes the coupling strategy used. The individual codes chosen, **FEFL098** for the fluid, **COSMIC-NASTRAN**, **DYNA3D** for the solid region, and **COSMIC-NASTRAN** for the heat transfer in solids, are briefly described in Section 3. Sections 4-6 discuss surface to surface interpolation, surface tracking and load transfer techniques. In Section 7, some demonstration runs are shown. Finally, conclusions and an outlook for future development are given in Section 8.

## 2. COUPLING ALGORITHM

When trying to compare the possible coupling algorithms, it is useful to start from the basic discrete equation systems obtained for the thermal, solid and fluid regions. In the following, we assume, without

loss of generality, that some form of spatial and temporal discretization has been carried out for the individual sub-disciplines and subdomains. The time-marching scheme is then cast into a system of equations for the increments of the unknowns in time, i.e. in so-called  $\Delta$ -form. For the **thermal field** in the solid region, we obtain a system of equations of the form:

$$\mathbf{LHST} \cdot \Delta \mathbf{T} = \left( \frac{1}{\Delta t} \mathbf{C} + \Theta \mathbf{K} \right) \cdot \Delta \mathbf{T} = \mathbf{f}_t^i + \mathbf{f}_t^e , \quad (1)$$

where

$\Delta t, \mathbf{C}, \mathbf{T}, \mathbf{K}, \mathbf{f}$  denote, respectively, the timestep, the heat capacitance matrix, nodal temperature vector, heat conduction matrix and the (internal and external) thermal loads vector, and we have lumped the left-hand-side matrix into the expression **LHST**. By splitting the degrees of freedom into those touching the fluid region ('f'), and the remaining ones ('t'), we obtain

$$\begin{Bmatrix} \mathbf{LHST}_{ff} & \mathbf{LHST}_{ft} \\ \mathbf{LHST}_{tf} & \mathbf{LHST}_{tt} \end{Bmatrix} \cdot \begin{pmatrix} \Delta \mathbf{T}_f \\ \Delta \mathbf{T}_t \end{pmatrix} = \begin{pmatrix} \mathbf{f}_f \\ \mathbf{f}_t \end{pmatrix}^i + \begin{pmatrix} \mathbf{f}_f \\ \mathbf{f}_t \end{pmatrix}^e + \begin{pmatrix} \mathbf{L} \cdot (\mathbf{q}_f + \Theta \Delta \mathbf{q}_f) \\ 0 \end{pmatrix} , \quad (2)$$

where the superscripts  $i, e$  denote internal (conduction) and external thermal loads respectively,  $\mathbf{L}$  is the load matrix and  $\mathbf{q}_f$  the vector of heat loads on the surface due to the fluid. For the **solid region**, we obtain a system of equations of the form:

$$\begin{aligned} \mathbf{LHSS} \cdot \Delta \dot{\mathbf{X}} &= (\alpha \mathbf{M}_s + \beta \mathbf{D} + \gamma \mathbf{K}) \Delta \dot{\mathbf{X}} = \\ \mathbf{f}_s^i + \mathbf{f}_s^e + \Theta \Delta \mathbf{f}_s^e + \mathbf{Q}(\mathbf{T} + \Theta \Delta \mathbf{T}) & , \quad (3) \end{aligned}$$

where  $\mathbf{M}_s, \mathbf{f}_s^i, \mathbf{f}_s^e, \mathbf{D}, \mathbf{K}, \mathbf{X}, \dot{\mathbf{X}}, \mathbf{Q}, \mathbf{T}$  denote, respectively, the mass-matrix, interal (stiffness, damping, inertia) and external force vector, damping matrix, stiffness matrix, displacement vector, velocity vector, thermal stress matrix and nodal temperatures,  $\alpha, \beta, \gamma, \Theta$  depend on the timestep and the timestepping scheme chosen, and we have lumped the left-hand-side matrix into the expression **LHSS**. By splitting the degrees of freedom into those touching the fluid region ('f'), and the remaining ones ('s'), we obtain

$$\begin{Bmatrix} \mathbf{LHSS}_{ff} & \mathbf{LHSS}_{fs} \\ \mathbf{LHSS}_{sf} & \mathbf{LHSS}_{ss} \end{Bmatrix} \cdot \begin{pmatrix} \Delta \dot{\mathbf{X}}_f \\ \Delta \dot{\mathbf{X}}_s \end{pmatrix} =$$

$$\begin{pmatrix} \mathbf{f}_f \\ \mathbf{f}_s \end{pmatrix}^i + \begin{pmatrix} \mathbf{f}_f \\ \mathbf{f}_s \end{pmatrix}^e + \begin{pmatrix} \mathbf{L} \cdot (\mathbf{s}_f + \Theta \Delta \mathbf{s}_f) \\ 0 \end{pmatrix} + \begin{pmatrix} \mathbf{Q}_f(\mathbf{T} + \Theta \Delta \mathbf{T}) \\ \mathbf{Q}_s(\mathbf{T} + \Theta \Delta \mathbf{T}) \end{pmatrix} , \quad (4)$$

where the superscripts  $i, e$  denote internal (stiffness, damping, inertia) and external forces respectively,  $\mathbf{L}$  is the load matrix and  $\mathbf{s}_f$  the fluid stresses (pressures, shear stresses) on the surface. For the **fluid region**, we obtain a system of equations of the form:

$$\mathbf{LHSF} \cdot \Delta \mathbf{U} = \left( \frac{1}{\Delta t} \mathbf{M}_f + \theta_f \mathbf{J} \right) \Delta \mathbf{U} = \mathbf{f}^i + \mathbf{f}^e , \quad (5)$$

where  $\mathbf{M}_f, \mathbf{J}, \mathbf{f}, \mathbf{U}$  denote, respectively, the mass matrix, Jacobian of the discretized fluxes, internal and external forces, and we have lumped the left-hand-side matrix into the expression **LHSF**. By splitting the degrees of freedom into those touching the solid region ('s'), and the remaining ones ('f'), we obtain

$$\begin{Bmatrix} \mathbf{LHSF}_{ss} & \mathbf{LHSF}_{sf} \\ \mathbf{LHSF}_{fs} & \mathbf{LHSF}_{ff} \end{Bmatrix} \cdot \begin{pmatrix} \Delta \mathbf{U}_s \\ \Delta \mathbf{U}_f \end{pmatrix} = \begin{pmatrix} \mathbf{f}_s \\ \mathbf{f}_f \end{pmatrix}^i + \begin{pmatrix} \mathbf{f}_s \\ \mathbf{f}_f \end{pmatrix}^e , \quad (6)$$

where the superscripts  $i, e$  denote the internal (viscous, advective, inertia) and external force vectors respectively. Before proceeding to the complete coupled equation system, we have to define the continuity of the variables across boundaries or domains. The temperatures required by the structure for thermal stress calculation are obtained from the CTD solver via interpolation:

$$\mathbf{T}_s = \mathbf{I}_{s,t} \mathbf{T}_t , \quad (7)$$

where  $\mathbf{I}_{s,t}$  is a 3-D interpolation matrix, which reverts back to the identity matrix in case that the nodes of the grids for structural and thermal analysis are coincident. In the same way, the temperatures at the 'wetted surfaces' of the CFD and CTD domains have to be the same:

$$\mathbf{T}_f = \mathbf{I}_{f,t} \mathbf{T}_t . \quad (8)$$

Here  $\mathbf{I}_{f,t}$  is now a surface to surface interpolation operator. The velocities at the surface of the structure and fluid domain are required to be the same, hence:

$$\mathbf{v}_f|_{\Gamma_s} = \mathbf{I}_{f,s} \mathbf{v}_s = \mathbf{I}_{f,s} \dot{\mathbf{X}}_s . \quad (9)$$

Finally, the thermal and mechanical loads the fluid exerts on the structure have to be transferred:

$$\mathbf{q}_f = \mathbf{G}_{tf} \mathbf{U}_f + \mathbf{G}_{ts} \mathbf{U}_s , \quad \mathbf{s}_f = \mathbf{G}_{sf} \mathbf{U}_f + \mathbf{G}_{ss} \mathbf{U}_s , \quad (10)$$

where we have used  $\mathbf{G}$  to indicate the gradient- or derivative-based operators involved.

The final coupled system then takes the form:

$$\left\{ \begin{array}{ccc}
 \text{LHST}_{ff} & \text{LHST}_{ft} & 0 \\
 \text{LHST}_{tf} & \text{LHST}_{tt} & 0 \\
 -\Theta Q_f \mathbf{I}_{ft} & -\Theta Q_f \mathbf{I}_{ft} & \text{LHSS}_{ff} \\
 -\Theta Q_s \mathbf{I}_{ft} & -\Theta Q_s \mathbf{I}_{ft} & \text{LHSS}_{fs} \\
 0 & 0 & 0 \\
 0 & 0 & 0
 \end{array} \right\} \cdot \begin{pmatrix} \Delta \mathbf{T}_f \\ \Delta \mathbf{T}_t \\ \Delta \dot{\mathbf{X}}_f \\ \Delta \dot{\mathbf{X}}_s \\ \Delta \mathbf{U}_s \\ \Delta \mathbf{U}_f \end{pmatrix} = \\
 \begin{pmatrix} \mathbf{f}_f \\ \mathbf{f}_s \\ \mathbf{f}_f \\ \mathbf{f}_t \\ \mathbf{f}_f \\ \mathbf{f}_s \end{pmatrix}^i + \begin{pmatrix} \mathbf{f}_f \\ \mathbf{f}_s \\ \mathbf{f}_f \\ \mathbf{f}_t \\ \mathbf{f}_f \\ \mathbf{f}_s \end{pmatrix}^e + \left\{ \begin{array}{ccc}
 0 & 0 & 0 \\
 0 & 0 & 0 \\
 Q_f \mathbf{I}_{ft} & Q_f \mathbf{I}_{ft} & 0 \\
 Q_s \mathbf{I}_{ft} & Q_s \mathbf{I}_{ft} & 0 \\
 0 & 0 & 0 \\
 0 & 0 & 0
 \end{array} \right\} \cdot \begin{pmatrix} \mathbf{T}_f \\ \mathbf{T}_t \\ \dot{\mathbf{X}}_f \\ \dot{\mathbf{X}}_s \\ \mathbf{U}_s \\ \mathbf{U}_f \end{pmatrix} \quad (11)$$

Given this complete system, we can now define possible coupling algorithms.

a) Tight coupling: We denote by tight coupling the simultaneous update of all variables, including (and most notably) those at the fluid/structure/thermal interface. This implies solving the complete system given by Eqn.(11) in one step. The formulation allows for different grids in the CFD, CSD and CTD domains, but the reader should realize that the derivation of the proper projection and interpolation matrices can be tedious in 3-D. From a numerical point of view, choosing this approach leads, for most cases, to an increase in the condition number for the matrices to be solved, with the associated solution difficulties. From a practical point of view, choosing this approach requires an almost complete re-write of the CFD, CSD and CTD codes into one single coupled code. This implies a loss of modularity (different numerical schemes, different codes), as well as the inability to couple several CFD, CSD and CTD codes. Moreover, the 'trade-oriented' aspect of each of the individual codes is blurred or lost, with the associated extra expenses for retraining the user base.

b) Loose coupling: We denote by loose coupling the separate update of the CFD, CSD and CTD domains, with a transfer of variables at the interface or the domain. The most common way of realizing this approach is by selecting a 'master surface' for a certain variable, and interpolating or projecting the variable to the other domain at the beginning of the next timestep. For CFD/CSD/CTD problems, the most natural combination is to select the CSD surface location and velocity as the 'master-grid' for displacements and temperatures, and the CFD grid as the 'master-grid' for the loads (pressures, shear-stresses, heat fluxes). The product of displacement times load yields work, making the combination physically appealing. This approach may be regarded as

an iterative solution of the combined system given by Eqn.(11). Each iterative pass is composed of the following steps (see Figure 3):

- Solve for CTD with imposed  $\mathbf{q}_f$ ;
- Solve for CSD with imposed  $\mathbf{s}_f, \mathbf{T}$ ;
- Solve for CFD with imposed  $\mathbf{T}_s, \mathbf{X}_s, \dot{\mathbf{X}}_s$ .

This procedure may be refined to include a transfer of the left-hand side Jacobians.

Depending on the time integration scheme used for the CTD, CSD and CFD domains, several simplifying strategies can be employed. Should explicit time integration be appropriate to advance the CTD, CSD and CFD regions (as is the case for some of the problems considered here), the loose and tight coupling systems are almost identical, the only error being the mass of fluid for the first row of elements adjacent to the solid. Should implicit time integration be best way to advance the CSD and CFD regions (as is the case for low-frequency aeroelastic applications), the LHS of the time-discrete form of Eqn.(11) will contain entries of the Jacobians of  $\mathbf{f}^i$ . In this case, the iterative strategy discussed above will have to be used for the loose coupling approach if equivalency with the tight coupling system is to be achieved. Finally, if only a steady-state solution for the coupled fluid-structure system is sought, the loose coupling approach may be used either with explicit or implicit time integration for the CTD, CSD and CFD domains without incurring any errors.

The variables on the boundaries are transferred back and forth between the different codes by a master code that directs the multi-disciplinary run. Two possible implementations are possible here:

- a) Single Executable: Each 'discipline code' (CFD, CSD, CTD, ..) is seen as a subroutine or object called by the master code

b) Multiple Executables: Each 'discipline code' (CFD, CSD, CTD, ..) runs as an independent process that awaits commands and data from a master process or other processes. The master process synchronizes the start and interruption of the different processes, and in particular the transfer of information.

In both cases, the transfer of geometrical and physical information is performed between the different codes without affecting their layout, basic functionality, and coding styles. This is seen as the main advantage of this approach.

A tremendous amount of man-years has been devoted to CFD, CSD and CTD codes, incorporating into them all the minor features that make these codes efficient, practical, user-friendly tools. The central assumption made here is that these codes will not be rewritten, should not be hampered in their present and future development, and nevertheless can be combined efficiently to solve strongly coupled CFD/CSD/CTD problems.

For structures that break, rupture, or deform markedly due to the loads exerted by the fluid, the corresponding CFD and/or CSD grid will require some form of remeshing, which can be either local or global in nature. If this remeshing can not be done automatically, the usefulness of such an approach will always remain limited. Therefore, automatic gridding techniques are an enabling technology for this class of problems. The CFD code employed here has, as one of its salient features, an automatic remeshing capability. This capability is very important for the class of fluid-structure interaction problems considered, and will be demonstrated in the examples shown below.

### 3. CODES SELECTED

The selection of the respective CFD/CSD/CTD codes was made according to the following guidelines:

- The code must be well proven;
- The code must be benchmarked;
- The code must be supported;
- The code must have a user base/community;

The three candidate codes chosen were: **FEFLO98** for the fluid, **COSMIC-NASTRAN** for linear structures **DYNA3D** for the solid. A brief overview of the physics being modelled, the numerical techniques employed, as well as useful engineering, meshing and software options available in these two codes is given in the sequel.

#### 3.1 CFD CODE: **FEFLO98**

a) Physics: **FEFLO98** is a simulation code for compressible and incompressible flows. The equations solved are the Euler, Laminar or Reynolds-averaged

Navier-Stokes equations, as well as the linear acoustics equations. The turbulence models available are the Smagorinsky, Baldwin-Lomax and  $k - \epsilon$  models, as well as a user-input option via subroutine. Equations of state supported by **FEFLO98** include ideal polytropic gas, real air EOS table look-up, water EOS table look-up, the JWL EOS for high explosives and a link to the general SESAME library of EOS. In order to handle situations with moving bodies and/or moving grids, the equations are solved in the Arbitrary Lagrangean-Eulerian frame [Don82].

Flows with particles are treated via a second solid phase. The particles interact with the fluid, exchanging mass, momentum and energy, and are integrated in a time-consistent manner with the fluid.

b) Numerics: The spatial discretization is accomplished via finite element techniques on unstructured tetrahedral grids. In order to achieve high execution speeds, edge-based data structures are used. Both central and upwind flux (van Leer [vLe74], Roe [Roe81], AUSM+ [Lio95]) formulations are used. For the temporal discretization, both Taylor-Galerkin and Runge-Kutta time integration schemes are possible. Monotonicity of the solution may be achieved through a blend of second and fourth order dissipation [Jam93], pressure-based, Flux-Corrected Transport (FCT) [Löh87], or classic TVD limitors. The particles are integrated using a second-order Runge-Kutta scheme, and optimal tracking techniques [Löh90, Siv94] have been implemented to expedite the transfer of information between fields and particles.

c) Engineering: In order to handle situations with moving bodies, **FEFLO98** offers a variety of options: prescribed motion, 6-DOF integration based on aerodynamic forces, and link to CSD codes.

A variety of boundary conditions can be prescribed to simulate as faithfully as possible engineering flows: sub-, tran-, and supersonic in/outflow, total pressure inflow b.c., static pressure, mach-number and normal flux outflow b.c., porous walls, and periodicity. At the same time, a large variety of diagnostics is produced by the code to track or display specific parts of the flowfield that are of special interest: 0-D probes (e.g. for station time history), 1-D line segments for x/y display, 2-D planes or iso-surfaces for contouring, flux trough surfaces, force and moment data on surfaces or bodies, on-line display of the flowfield, etc.

d) Meshing Options: **FEFLO98** allows for automatic adaptive h-refinement [Löh92] and automatic remeshing [Löh88,90] in order to enhance the solution accuracy, even for situations with moving bodies.

e) Software: **FEFLO98** is written in FORTRAN-77 and fully vectorized. Renumbering techniques [Löh93,97] are used extensively in order to

avoid cache-misses on RISC-based machines and cache-line contingencies on shared memory parallel machines. For distributed memory parallel machines domain splitting [Löh93] and message passing are employed. The code runs on all major workstations, vector-supercomputers and parallel platforms.

**FEFLO98** is a well-proven and benchmarked code used extensively by the authors and others in the CFD community [Bau93,94,95,96,98].

### 3.2 Linear CSD CODE: **COSMIC-NASTRAN**

a) Physics: **COSMIC-NASTRAN** is a simulation code especially suited for solids undergoing elastic deformations. In addition, **COSMIC-NASTRAN** may be used to solve any linear second order elliptic or parabolic PDE, e.g. Laplace, Poisson and Helmholtz equations, time-dependent linear heat conductances, etc. The conservation equations for momentum are written and solved for in the Lagrangian frame of reference. The small deformation, small strain formulation is employed throughout. The code incorporates isotropic or orthotropic elastic material models, and many variations for composites [Mac81].

b) Numerics: The spatial discretization is accomplished via finite element techniques on unstructured grids. A very large number of different element types is supported (it is rumored that **COSMIC-NASTRAN** is over two million lines long !), among them truss beam elements, several triangular and quadrilateral shell elements, and tetrahedral, prismatic and hexahedral solid elements. The temporal discretization is carried out using an implicit difference method, which is unconditionally stable. The resulting equation systems for static, dynamic and eigenvalue analysis are solved directly, using variants of Gaussian solution techniques. **COSMIC-NASTRAN** incorporates elaborate bandwidth minimization algorithms to reduce the cost of direct solvers.

c) Engineering: Given its large history as the *de facto* standard code for linear structural mechanics, **COSMIC-NASTRAN** incorporates a very large number of convenient features that have proven useful for the efficient modeling of engineering problems. A large variety of diagnostics is produced by the code to track or display specific parts of the structure that are of special interest: 0-D probes (e.g. for station time history), 1-D line segments for x/y display, 2-D planes or iso-surfaces for contouring, stress, strain, force and moment data on surfaces or fields, etc.

d) Meshing Options: **COSMIC-NASTRAN** in its current state does not allow for automatic adaptive h-refinement or automatic remeshing.

e) Software: **COSMIC-NASTRAN** is written in FORTRAN-66. The code was written before the advent of vector or parallel machines CRAY-type machines. Given that the number of degrees for linear structures are typically only a fraction of those used for flow simulations, this is not a major drawback. **COSMIC-NASTRAN** has been ported to all major platforms, from PC's to parallel platforms.

**COSMIC-NASTRAN** is a well-proven, benchmarked code used extensively in industry, government laboratories and academia.

### 3.3 Non-Linear CSD CODE: **DYNA3D**

a) Physics: **DYNA3D** is a simulation code especially suited for solids undergoing rapid and severe deformation. The conservation equations for momentum are written and solved for in the Lagrangian frame of reference. The large deformation, large strain formulation is employed throughout. The code incorporates forty-one different material models, among them linear elastic, linear elastic-plastic, strain-rate sensitive steel with fracture, hardening material models, a geological cap model for soil materials, and a variety of concrete models that simulate fracturing behavior [Whi93]. **DYNA3D** also offers eleven equations of state models, including equations of state for high explosives.

b) Numerics: The spatial discretization is accomplished via finite element techniques on unstructured grids. The elements available for structural modelling are one truss and two beam elements, several quadrilateral shell elements (e.g. Belytschko-Tsai [Bel84], Hughes-Liu [Hug81], YASE [Whi93], and QPH [Bel94], and hexahedral elements with one-point integration for the 3-D solids. The shells allow for multiple integration points across the thickness, making it possible to accurately treat nonlinear plastic behavior of simple and composite shells. Several hourglass control options may be used. We have found that the Flanagan-Belytschko hourglass control [Fla81] works best for the unstructured hexahedral grids we most often employ. The temporal discretization is carried out using an explicit central difference method, which is conditionally stable.

c) Engineering: **DYNA3D** incorporates a large number of convenient features that prove especially useful for realistic engineering problems. The following is a non-exhaustive list of those features that were particularly relevant to our class of applications. The user may prescribe non-reflecting boundary conditions which eliminate stress wave reflections at model boundaries, making it possible to use smaller models. There are twelve types of sliding-interface algorithms to treat different interface conditions between interacting parts. Sliding-interface algorithms permit

the treatment of contact conditions with friction, gap opening, spotwelds, etc. For civil engineering applications, there are rebar-concrete interaction algorithms which include degradation and failure of bond.

A large variety of diagnostics is produced by the code to track or display specific parts of the structure that are of special interest: 0-D probes (e.g. for station time history), 1-D line segments for x/y display, 2-D planes or iso-surfaces for contouring, stress, strain, force and moment data on surfaces or fields, etc.

d) Meshing Options: **DYNA3D** in its current state does not allow for automatic adaptive h-refinement or automatic remeshing. Work is currently in progress to incorporate these feature into **DYNA3D** .

e) Software: **DYNA3D** is written in FORTRAN-77 and fully vectorized. The code was written with CRAY-type machines in mind, but runs well on all major workstations, vector-supercomputers and some parallel platforms. It employs dynamic memory allocation, making it capable of solving very large problems.

**DYNA3D** is a well-proven and benchmarked code used extensively by the authors and others in the CSD community [Gou82, Cha82, Whi93]. **DYNA3D** was developed at the Lawrence Livermore National Laboratories by Dr. John Hallquist with contributions from Dr. David Benson and Dr. Robert Whirley. **DYNA3D** has been successfully used for a large number of applications, including nuclear and conventional weapon design, car and airplane crashworthiness studies, analysis of reinforced structures such as bunkers, tunnels, and silos, as well as spent nuclear shipping cases. It is supported and maintained by Lawrence Livermore National Laboratories.

#### 4. SURFACE TO SURFACE INTERPOLATION

One of the main aims of the proposed approach is to couple the different codes in such a way that each one of the codes used is modified in the least possible way. Moreover, the option of having different grids for different disciplines (CFD/CSD/CTD/CEM..), as well as adaptive grids that vary in time, implies that in most cases no fixed common variables will exist at the boundaries. Therefore, fast and accurate interpolation techniques are required. As the grids may be refined/coarsened during timesteps, and the surface deformations may be severe, the interpolation procedures have to combine speed with generality. Algorithmic aspects of procedures that fulfill these requirements have been discussed in [Knu73, Sed83, Löh95] (volume) and [Löh95, Mam95, Ceb97] (surface), and do not need to be repeated here.

#### 4.1 Unwrapping of Doubly Defined Faces

Consider the common case of thin structural elements, e.g. roofs, walls, stiffeners, etc. surrounded by a fluid medium. The structural elements will be discretized using shell elements. These shell elements will be affected by loads from both sides. Most CSD codes require a list of faces on which loads are exerted. This implies that the shell elements loaded from both sides will appear twice in this list. In order to be able to incorporate thickness and interpolate between CSD and CFD surface grids in a unique way, these doubly defined faces are identified, and new points are introduced. The first step is to identify the doubly defined faces. A linked list that stores the faces surrounding each point is constructed. Each face is then checked by performing an exhaustive comparison of the points of each of the faces surrounding the first node of each face. This will identify doubly defined faces in  $O(N_f)$  complexity ,where  $N_f$  is the number of faces. Should this check reveal the existence of doubly defined faces, new points are introduced using an unwrapping procedure [Löh95].

#### 5. SURFACE TRACKING TECHNIQUES

An important question that needs to be addressed is how to make the different grids follow one another when deforming surfaces are present. Consider the typical aeroelastic case of a wing deforming under aerodynamic loads. For accuracy purposes, the CFD discretization will be fine on the surface, and the surface will be modelled as accurately as possible from the CAD-CAM data at the start of the simulation. On the other hand, a CSD discretization that models the wing as a series of plates may be entirely appropriate. If one would force the CFD surface to follow the CSD surface, the result would be a wing with no thickness, clearly inappropriate for an acceptable CFD result. On the other hand, for strong shock/object interactions with large plastic deformations and possible tearing, forcing the CFD surface to follow exactly the CSD surface is the correct way to proceed. These two examples indicate that more than one strategy may have to be used to interpolate and move the surface of the CFD mesh as the structure moves. We have incorporated the following techniques:

a) Exact tracking with linear interpolation. This is the most straightforward case, but, as could be seen from the example described above, may lead to bad results.

b) Exact tracking with quadratic interpolation. In this case, the surface normals are recovered at the end-points of the surface triangulation. For each edge of the triangulation, the midpoint is extrapolated using a Hermitian polynomial [Löh95]. In this

way, quadratic triangles are obtained. The surface is then approximated/interpolated using this higher order surface.

c) Tracking with initial distance vector. In many instances, e.g. thick shells, the CFD and CSD domains will never coincide. A way to circumvent this dilemma is to compute the difference vector between the initial CSD and CFD surfaces, and maintain this vector (allowing for translation and rotation) for the duration of the coupled run. Several options are possible here, e.g. surface normals, point normals, etc. [Ceb97].

An important area still under investigation is how to handle, in an efficient and automatic way, models that exhibit incompatible dimensionalities. An example for such a 'reduced model' would be an aerothermoelectric problem where the wing structure is modeled by a torsional beam (perfectly acceptable for the lowest eigenmodes), the fluid by a 3-D volumetric mesh, and the heat transfer by a network model. It is easy to see that the proper specification of movement for the CFD surface based on the 1-D beam, as well as the load transfer from the fluid to the beam and network points represent non-trivial problems for a general, user-friendly computing environment.

## 6. CFD-CSD/CTD LOAD TRANSFER

During each iteration, the CFD loads have to be transferred to the CTD/CSD grids. Simple pointwise interpolation can be employed for those cases in which the elements of the CTD/CSD surface meshes are smaller or of similar size than the elements of the CFD surface mesh. However, this approach is not conservative, and will not yield accurate results for the common case of CTD/CSD surface elements being larger than their CFD counterpart. Considering without loss of generality the pressure loads only, it is desirable to attain:

$$p_s(\mathbf{x}) \approx p_f(\mathbf{x}) , \quad (12)$$

while being conservative in the sense of:

$$\mathbf{f} = \int p_s \mathbf{n} d\Gamma = \int p_f \mathbf{n} d\Gamma , \quad (13)$$

where  $p_f, p_s$  denote the pressures on the fluid and solid material surfaces, and  $\mathbf{n}$  is the normal vector. These requirements may be combined by employing a weighted residual method. With the approximations:

$$p_s = N_s^i p_{is} , \quad p_f = N_f^j p_{jf} , \quad (14)$$

we have

$$\int N_s^i N_s^j d\Gamma p_{is} = \int N_s^i N_f^j d\Gamma p_{jf} , \quad (15)$$

which may be rewritten as:

$$\mathbf{M} \mathbf{p}_s = \mathbf{r} = \mathbf{L} \mathbf{p}_f . \quad (16)$$

Here  $\mathbf{M}$  is a 'consistent mass-matrix', and  $\mathbf{L}$  a 'loading matrix'. The solution of this coupled system of equations is obtained iteratively in the now familiar way:

$$\mathbf{M}_l \cdot (\mathbf{p}_s^{i+1} - \mathbf{p}_s^i) = \mathbf{r} - \mathbf{M} \cdot \mathbf{p}_s^i , \quad (17)$$

where  $\mathbf{M}_l$  is the 'lumped mass matrix'. Typically, three iterations are sufficient to achieve an accurate result. One can also show that Eqn.(16) is equivalent to the least-squares minimization of

$$I = \int [p_s - p_f]^2 d\Gamma , \quad (18)$$

We remark that the weighted residual method is conservative in the sense of Eqn.(13). The sum of all shape-functions at any given point is unity ( $\sum_i N_s^i(\mathbf{x}) = 1$ ), and therefore:

$$\begin{aligned} \int p_s d\Gamma &= \int N_s^i d\Gamma p_{is} = \int \sum_i N_s^i N_s^j d\Gamma p_{is} , \\ &= \sum_i \int N_s^i N_s^j d\Gamma p_{is} = \sum_i \int N_s^i N_f^j d\Gamma p_{jf} \\ &= \int \sum_i N_s^i N_f^j d\Gamma p_{jf} = \int N_f^j d\Gamma p_{jf} = \int p_f d\Gamma \end{aligned} \quad (19)$$

The most problematic part of the weighted residual method is the evaluation of the integrals appearing on the right-hand side of Eqn.(15). When the CFD and CSD surface meshes are not nested, this is a formidable task. The adaptive Gaussian quadrature procedure proposed in [Ceb97] has proven very successful for the evaluation of these integrals.

## 7. EXAMPLE RUNS

7.1 Generic Weapon Fragmentation: This second example shows a fully coupled CFD/CSD run. The structural response was calculated using **GADYNA** [Pel95, Pel97, Pel98], an enhanced version of **DYNA3D** that incorporates adaptive refinement and improved contact algorithms. The structural elements were assumed to fail once the average strain in an element exceeded 60%. At the beginning, the CFD domain consisted of two separate regions. These regions connected as soon as fragmentation started. In order to handle narrow gaps during the break-up process, the failed CSD elements were shrunk by a fraction of their size. This alleviated the timestep constraints imposed by small elements without affecting the overall accuracy. The final breakup led to approximately 1200 objects in the flowfield. Figures 4,5 show

the fluid pressure and CSD surface velocity at three different times during the simulation. Typical meshes for this simulation were of the order of 8.0 Mtets, and the simulations required of the order of 50 hours on the SGI Origin 2000 running on 32 processors.

**7.2 Nose-Cone:** As a second example, we consider a generic nose-cone. The cone is 1 m long and its angle is approximately  $20^\circ$ . The freestream conditions were set as follows: density  $\rho = 1.25 \text{ kg/m}^3$ , velocity  $v = 10^3 \text{ m/sec}$ , pressure  $p = 10^5 \text{ N/m}^2$ , free stream viscosity  $\mu = 1.8 \cdot 10^{-5} \text{ kg/m/sec}$ , initial temperature  $T = 273 \text{ K}$ , heat coefficient at constant pressure  $c_p = 1000$ , polytropic coefficient  $\gamma = 1.4$ , Prandtl-nr. of  $Pr = 0.71$  and angle of attack of  $\alpha = 10^\circ$ . This yields a Reynolds-nr. of approximately  $Re = 7 \cdot 10^7$  based on the length of the cone, and a Mach-nr. of  $M_\infty = 3.0$ . The effects of turbulence were simulated using the Baldwin-Lomax turbulence model. Sutherland's law was assumed for the dependence of viscosity and conductivity on temperature. A suitable grid with highly stretched elements in the boundary layer was generated using the advancing layers/ advancing front technique. This CFD grid had approximately 574 Ktet elements. The actual shell thickness of the cone was assumed be  $\delta = 1 \text{ mm}$ , with three reinforcement plates of  $\delta = 2 \text{ mm}$ . The material properties for the shells were as follows: Young's modulus  $E = 2 \cdot 10^9 \text{ N/m}^2$ , Poisson's ratio  $\nu = 0.30$ , density  $\rho = 7.84 \cdot 10^3 \text{ kg/m}^3$ , and thermal expansion coefficient  $\beta = 1.25 \cdot 10^{-5} \text{ 1/K}$ . The structure was discretized with the triangular shell element CTRIA2. The same discretization was also used for the calculation of the temperature field. The cone was assumed to be clamped at the base; also, at this location, the temperatures were assumed to be fixed. The solution was initiated by converging the fluid-thermal problem, without any structural deformation. This took approximately 10 iterations. Thereafter, the fluid-structure-thermal problem was solved in only 2 iterations. Figures 6,7 show grids and the results obtained.

**7.3 Deforming Panel:** As a second example, we consider supersonic flow over a deforming panel, as proposed by Thornton [Tho88]. A diagram of the initial conditions is shown in Figure 8. The panel is 0.1016 m long and 0.00254 m thick. The freestream conditions were set as follows: density  $\rho = 1.25 \text{ kg/m}^3$ , velocity  $v = 1.42 \cdot 10^3 \text{ m/sec}$ , pressure  $p = 10^5 \text{ N/m}^2$ , free stream viscosity  $\mu = 1.8 \cdot 10^{-5} \text{ kg/m/sec}$ , initial temperature  $T = 273 \text{ K}$ , heat coefficient at constant pressure  $c_p = 1000$ , polytropic coefficient  $\gamma = 1.4$  and Prandtl-nr. of  $Pr = 0.71$ . This yields a freestream Mach-nr. of  $M_\infty = 4.26$ . The effects of turbulence were simulated using the Baldwin-Lomax turbulence model, and a 1/7th profile was assumed at inflow. Sutherland's law was assumed for the dependence of viscosity and conductivity on temperature. A suit-

able grid with highly stretched elements in the boundary layer was generated using the advancing layers/ advancing front technique. This CFD grid had approximately 174 Ktet elements. The material properties for the panel were as follows: Young's modulus  $E = 2 \cdot 10^9 \text{ N/m}^2$ , Poisson's ratio  $\nu = 0.30$ , density  $\rho = 7.84 \cdot 10^3 \text{ kg/m}^3$ , and thermal expansion coefficient  $\beta = 1.25 \cdot 10^{-5} \text{ 1/K}$ . The structure was discretized with hexahedral volume elements CIHEX1. The same discretization was also used for the calculation of the temperature field. The plate was assumed to be clamped at the base extremities and adiabatic walls were assumed throughout, except at the contact of panel and fluid. The final result was obtained after 5 fluid-structure-thermal iterations. Figure 9 shows the surface of the CFD grid and the results obtained.

## 8. CONCLUSIONS AND OUTLOOK

A fluid/structure/thermal interaction algorithm based on the loose coupling of production CFD, CSD and CTD codes has been described. The algorithm allows a cost-effective re-use of existing software, with a minimum amount of alterations required to account for the interaction of the different media. Several example runs using **FEFLO98** as the CFD code, and **COSMIC-NASTRAN** or **DYNA3D** as the CSD/CTD code, demonstrate the effectiveness of the proposed methodology.

Future developments will center on:

- Treatment of reduced models, or models with incompatible dimensionalities;
- Improved reliability for complex geometries undergoing severe deformations, especially when contact is present;
- Extensions to other multidisciplinary problems, including electromagnetic loads; and
- Extensions across different spatio/temporal scales, i.e. the coupling of micro-physics codes to these macro-physics codes.

## 9. ACKNOWLEDGMENTS

This work was partially supported by DNA and AFOSR, with Drs. Mike Giltrud and Leonidas Sakell as the technical monitors.

## 10. REFERENCES

- [Ada98] P. Adamidis, A. Beck, U. Becker-Lengau, Y. Ding, M. Franzke, H. Holthoff, M. Laux, A. Müller, M. Münch, A. Reuter, B. Steckel and R. Tilch - Steel Strip Production - A Pilot Application for Coupled Simulation With Several Calculation Systems; *Metal Forming 98*, Birmingham, UK (1998).
- [Alo95] J. Alonso, L. Martinelli and A. Jameson - Multi-grid Unsteady Navier-Stokes Calculations with Aeroelastic Applications; *AIAA-95-0048* (1995).

[Bat88] J.T. Batina, R.M. Bennet, D.A. Seidel, H.J. Cunningham and S.R. Bland - Recent Advances in Transonic Computational Aeroelasticity; *Comp. Struct.* 30, No.1/2, 29-37, (1988).

[Bau93] J.D. Baum, H. Luo and R. Löhner - Numerical Simulation of a Blast Inside a Boeing 747; *AIAA-93-3091* (1993).

[Bau94] J.D. Baum, H. Luo and R. Löhner - A New ALE Adaptive Unstructured Methodology for the Simulation of Moving Bodies; *AIAA-94-0414* (1994).

[Bau95] J.D. Baum, H. Luo and R. Löhner - Numerical Simulation of Blast in the World Trade Center; *AIAA-95-0085* (1995).

[Bau96] J.D. Baum, H. Luo, R. Löhner, C. Yang, D. Pelessone and C. Charman - A Coupled Fluid/Structure Modeling of Shock Interaction with a Truck; *AIAA-96-0795* (1996).

[Bau98] J.D. Baum, H. Luo and R. Löhner - The Numerical Simulation of Strongly Unsteady Flows With Hundreds of Moving Bodies; *AIAA-98-0788* (1998).

[Bel84] T. Belytschko and Tsay - Explicit Algorithm for Nonlinear Dynamics of Shells; *Comp. Meth. Appl. Mech. Eng.* 43, 251-276 (1984).

[Bel94] T. Belytschko and I. Leviathan - Physical Stabilization of the 4-Node Shell Element with One Point Quadrature; *Comp. Meth. Appl. Mech. Eng.* 113, 321-350, (1994).

[Bos93] A.H. Boschitsch and T.R. Quackenbush - High Accuracy Computations of Fluid-Structure Interaction in Transonic Cascades; *AIAA-93-0485* (1993).

[Ceb97] J.R. Cebral and R. Löhner - Conservative Load Projection and Tracking for Fluid-Structure Problems; *AIAA J.* 35, 4, 687-692 (1997).

[Ceb97] J.R. Cebral and R. Löhner - Fluid-Structure Coupling: Extensions and Improvements; *AIAA-97-0858* (1997).

[Ceb98] J.R. Cebral and R. Löhner - Interactive On-Line Visualization and Collaboration for Parallel Unstructured Multidisciplinary Applications; *AIAA-98-0077* (1998).

[Cha82] C.M. Charman, R.M. Grenier, and R.R. Nickell - Large Deformation Inelastic Analysis of Impact for Shipping Casks; *Comp. Meth. Appl. Mech. Eng.* 33, 759-784, (1982).

[Coc97] COCOLIB Deliverable 1.1: Specification of the COUpling LIBRARY; CISPAR ESPRIT Project 20161, See <http://www.pallas.de/cispar/pages/docu.htm> (1997).

[Cod98] CSD codes such as NASTRAN, ANSYS, ABAQUS, MARC, NISA, ADINA, DYNA3D, PAM-CRASH, etc., CFD codes such as FLUENT, FIDAP, STAR-CD, RAMPANT, FEFLO, PAM-FLOW, etc., CTD codes such as NASTRAN, ANSYS, SINDA, etc.

[Don82] J. Donea - An Arbitrary Lagrangian-Eulerian Finite Element Method for Transient Dynamic Fluid-Structure Interactions; *Comp. Meth. Appl. Mech. Eng.* 33, 689-723 (1982).

[Eve90] G.C. Everstine and F.M. Henderson - Coupled Finite Element/Boundary Element Approach for Fluid-Structure Interaction; *J. Acoust. Soc. Am.* 87, 5, 1938-1947 (1990).

[Eve91] G.C. Everstine - Prediction of Low Frequency Vibrational Frequencies of Submerged Structures; *J. Vibrations and Acoustics* 113, (1991).

[Fel93] F.F. Felker - Direct Solution of Two-Dimensional Navier-Stokes Equations for Static Aeroelasticity Problems; *AIAA J.* 31, 1, 148-153 (1993).

[Fla81] D.P. Flanagan and T. Belytschko - A Uniform Strain Hexahedron and Quadrilateral with Orthogonal Hourglass Control; *Int. J. Num. Meth. Eng.* 17, 679-706, (1981).

[Gas87] J. Gaski and R.L. Collins - SINDA 1987-ANSI Revised User's Manual; Network Analysis Associates, Inc. (1987).

[Gou82] G.L. Goudreau and J.O. Hallquist - Recent Developments in Large-Scale Finite Element Lagrangean Hydrocode Technology; *Comp. Meth. Appl. Mech. Eng.* 33, 725-757 (1982).

[Gri98] GRISSLI - Numerical Simulation of Coupled Problems on Parallel Computers; BMBF-Project, Contract No. 01 IS 512 A-C/GRISSLI, Germany, See <http://www.gmd.de/SCAI/scicomp/grissli/> (1998).

[Gur90] G.P. Guruswamy - Unsteady Aerodynamic and Aerelastic Calculations for Wings Using Euler Equations; *AIAA J.* 28, 3, 461-469 (1990).

[Gur93] G.P. Guruswamy and C. Byun - Fluid-Structural Interactions Using Navier-Stokes Flow Equations Coupled with Shell Finite Element Structures; *AIAA-93-3087* (1993).

[Hug81] T.J.R. Hughes and W.K. Liu - Nonlinear Finite Element Analysis of Shells: Part I. Three Dimensional Shells; *Comp. Meth. Appl. Mech. Eng.* 26, 331-362 (1981).

[Jac87] P.S. Jackson and G.W. Christie - Numerical Analysis of Three-Dimensional Elastic Membrane Wings; *AIAA J.* 25, 5, 676-682, (1987).

[Jam93] A. Jameson - Artificial Diffusion, Upwind Biasing, Limiters and Their Effect on Accuracy and Multigrid Convergence in Transonic and Hypersonic Flows; *AIAA-93-3359* (1993).

[Knu73] D.N. Knuth - *The Art of Computer Programming*, Vol. 3; Addison-Wesley (1973).

[Les96] M. Lesoinne and Ch. Farhat - Geometric Conservation Laws for Flow Problems With Moving Boundaries and Deformable Meshes, and Their

Impact on Aeroelastic Computations; *Comp. Meth. Appl. Mech. Eng.* 134, 71-90 (1996).

[Lio95] M.-S. Liou - Progress Towards an Improved CFD Method: AUSM<sup>+</sup>; *AIAA-95-1701-CP* (1995).

[Löh87] R. Löhner, K. Morgan, J. Peraire and M. Vahdati - Finite Element Flux-Corrected Transport (FEM-FCT) for the Euler and Navier-Stokes Equations; *Int. J. Num. Meth. Fluids* 7, 1093-1109 (1987).

[Löh88] R. Löhner and P. Parikh - Three-Dimensional Grid Generation by the Advancing Front Method; *Int. J. Num. Meth. Fluids* 8, 1135-1149 (1988).

[Löh88] R. Löhner - An Adaptive Finite Element Solver for Transient Problems with Moving Bodies; *Comp. Struct.* 30, 303-317 (1988).

[Löh90] R. Löhner - Three-Dimensional Fluid-Structure Interaction Using a Finite Element Solver and Adaptive Remeshing; *Computer Systems in Engineering* 1, 2-4, 257-272 (1990).

[Löh90] R. Löhner and J. Ambrosiano - A Vectorized Particle Tracer for Unstructured Grids; *J. Comp. Phys.* 91, 1, 22-31 (1990).

[Löh92] R. Löhner and J.D. Baum - Adaptive H-Refinement on 3-D Unstructured Grids for Transient Problems; *Int. J. Num. Meth. Fluids* 14, 1407-1419 (1992).

[Löh93] R. Löhner - Some Useful Renumbering Strategies for Unstructured Grids; *Int. J. Num. Meth. Eng.* 36, 3259-3270 (1993).

[Löh93] R. Löhner, R. Ramamurti and D. Martin - A Parallelizable Load Balancing Algorithm; *AIAA-93-0061* (1993).

[Löh94] R. Löhner and J. McAnally - Transient and Steady Heat Conduction Using an Adaptive Finite Element CAD-Based Approach; *Int. J. Heat and Fluid Flow* 4, 311-327 (1994).

[Löh95] R. Löhner, C. Yang, J. Cebral, J.D. Baum, H. Luo, D. Pelessone and C. Charman - Fluid-Structure Interaction Using a Loose Coupling Algorithm and Adaptive Unstructured Grids; *AIAA-95-2259* [Invited] (1995).

[Löh95] R. Löhner - Robust, Vectorized Search Algorithms for Interpolation on Unstructured Grids; *J. Comp. Phys.* 118, 380-387 (1995).

[Löh97] R. Löhner - Renumbering Strategies for Unstructured- Grid Solvers Operating on Shared- Memory, Cache- Based Parallel Machines; *AIAA-97-2045-CP* (1997).

[Luo93] H. Luo, J.D. Baum, R. Löhner and J. Cabello - Adaptive Edge-Based Finite Element Schemes for the Euler and Navier-Stokes Equations; *AIAA-93-0336* (1993).

[Luo94] H. Luo, J.D. Baum and R. Löhner - Edge-Based Finite Element Scheme for the Euler Equations; *AIAA J.* 32, 6, 1183-1190 (1994).

[Mac81] R. MacNeal - NASTRAN Theoretical Manual; *NASA Scientific and Technical Information Office* (1981).

[Mam95] N. Maman and C. Farhat - Matching Fluid and Structure Meshes for Aeroelastic Computations: A Parallel Approach; *Computers and Structures* 54, 4, 779-785 (1995).

[Mes96] E. Mestreau and R. Löhner - Airbag Simulation Using

[Mes96] E. Mestreau and R. Löhner - Airbag Simulation Using Fluid/Structure Coupling; *AIAA-96-0798* (1996).

[Pel95] D. Pelessone and C.M. Charman - Adaptive Finite Element Procedure for Non-Linear Structural Analysis; *1995 ASME/JSME Pressure Vessels and Piping Conference*, Honolulu, Hawaii, July (1995).

[Pel97] D. Pelessone and C.M. Charman - An Adaptive Finite Element Procedure for Structural Analysis of Solids; *1997 ASME Pressure Vessels and Piping Conference*, Orlando, Florida, July (1997).

[Pel98] D. Pelessone and C.M. Charman - A General Formulation of a Contact Algorithm with Node/Face and Edge/Edge Contacts; *1998 ASME Pressure Vessels and Piping Conference*, San Diego, California, July (1998).

[Pro92] E.J. Probert, O. Hassan, K. Morgan and J. Peraire - Adaptive Explicit and Implicit Finite Element Methods for Transient Thermal Analysis; *Int. J. Num. Meth. Eng.* 35, 655-670 (1992).

[Rau93] R.D. Rausch, J.T. Batina and H.T.Y. Yang - Three-Dimensional Time-Marching Aerelastic Analyses Using an Unstructured-Grid Euler Method; *AIAA J.* 31, 9, 1626-1633 (1993).

[Roe81] P.L. Roe - Approximate Riemann Solvers, Parameter Vectors and Difference Schemes; *J. Comp. Phys.* 43, 357-372 (1981).

[Sed83] R. Sedgewick - *Algorithms*; Addison-Wesley (1983).

[Siv94] S. Sivier, E. Loth, J.D. Baum and R. Löhner - Unstructured Adaptive Remeshing Finite Element Method for Dusty Shock Flows; *Shock Waves* 4, 15-23 (1994).

[Tam97] K.K. Tam and R.R. Namburu - Computational Approaches With Applications to Non-Classical Thermomechanical Problems; *Appl. Mech. Rev.* 50, 9, 514-551 (1997).

[Tho88] E.A. Thornton and P. Dechaumphai - Coupled Flow, Thermal and Structural Analysis of Aerodynamically Heated Panels; *J. Aircraft* 25, 11, 1052-1059 (1988).

[vLe74] B. van Leer - Towards the Ultimate Conservative Scheme. II. Monotonicity and Conservation Combined in a Second Order Scheme; *J. Comp. Phys.* 14, 361-370 (1974).

[Whi93] R.G. Whirley and J.O. Hallquist - DYN3D, A

Nonlinear Explicit, Three-Dimensional Finite Element Code for Solid and Structural Mechanics - User Manual; UCRL-MA-107254, Rev.1, (1993).

[Zie88] O.C. Zienkiewicz and R. Taylor - *The Finite Element Method*; McGraw Hill (1988).

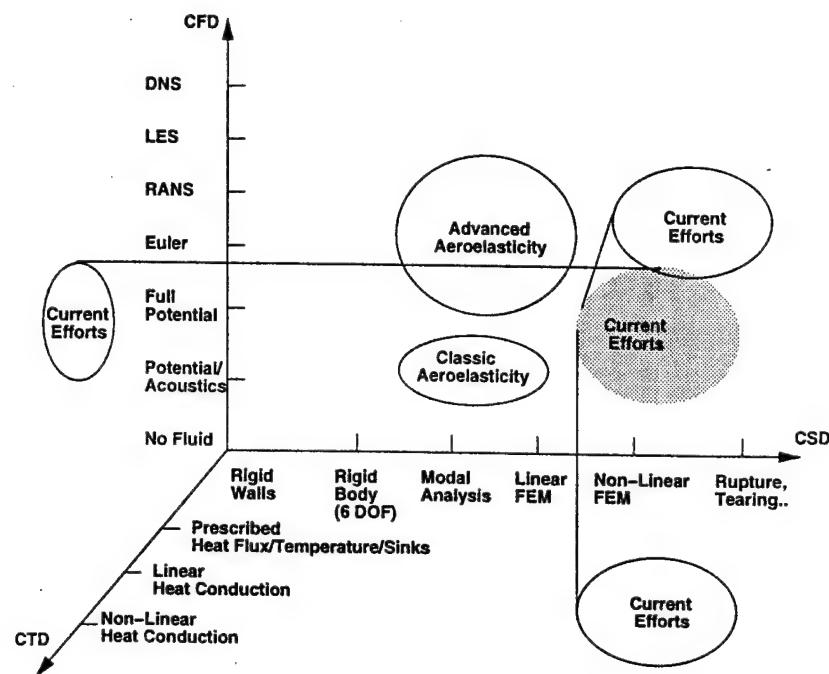


Figure 1 Fluid-Structure-Thermal Interaction

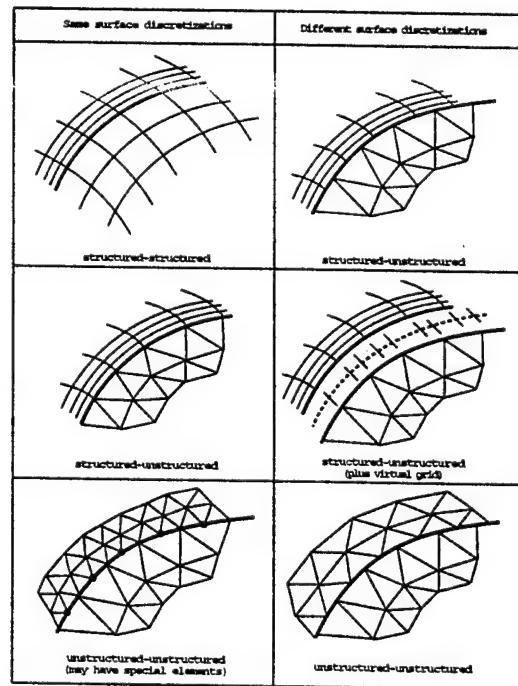


Figure 2 Coupling Different Discretizations

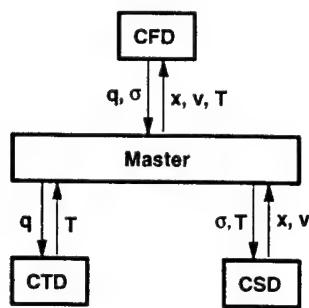


Figure 3 Loose Coupling

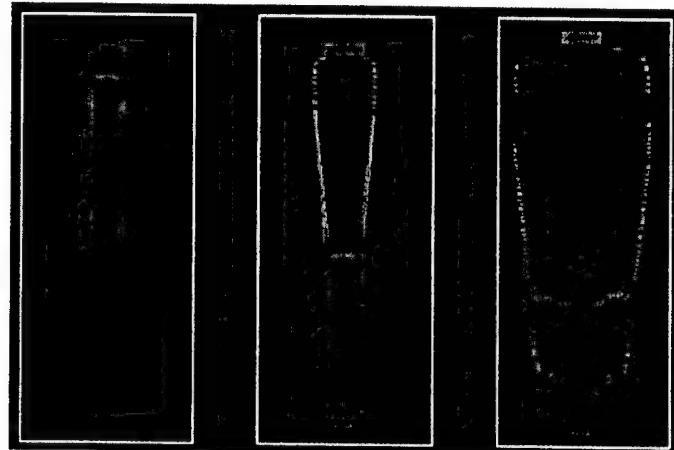


Figure 4 Generic Weapon Fragmentation: Pressure at Different Times



Figure 5 Generic Weapon Fragmentation: Surface Velocity at Different Times

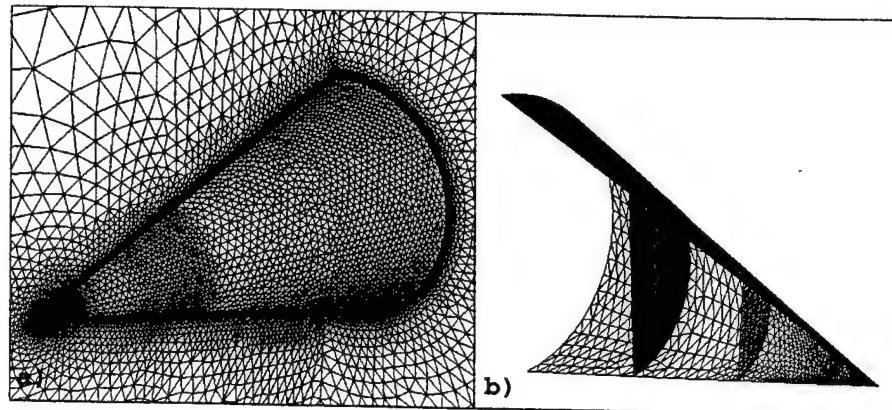


Figure 6 Nose-Cone: Surface Grids for CFD and CSD/CTD

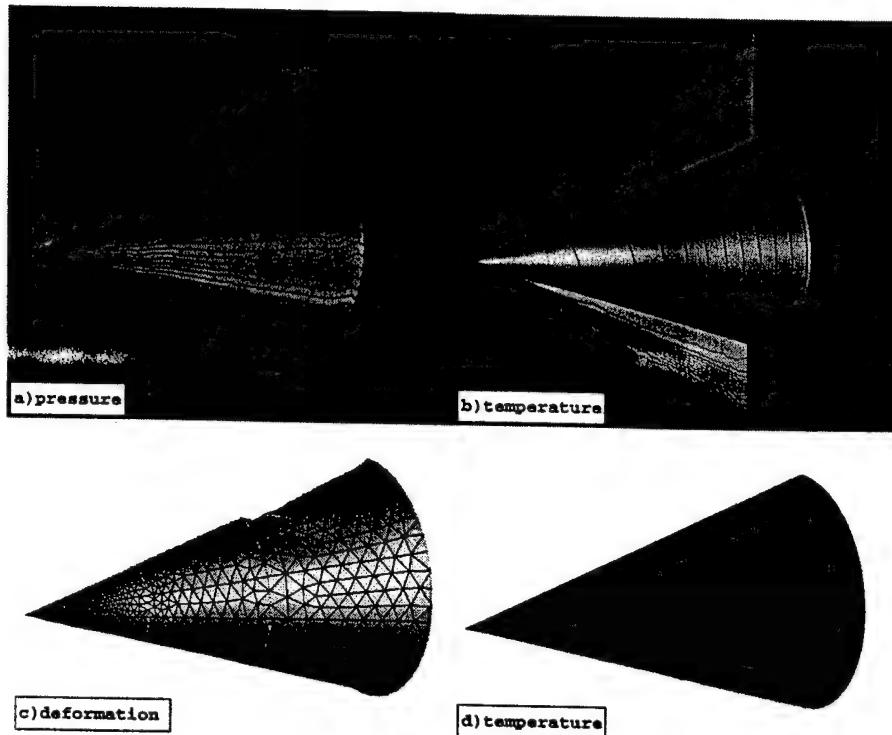


Figure 7 CFD/CSD/CTD Results Obtained

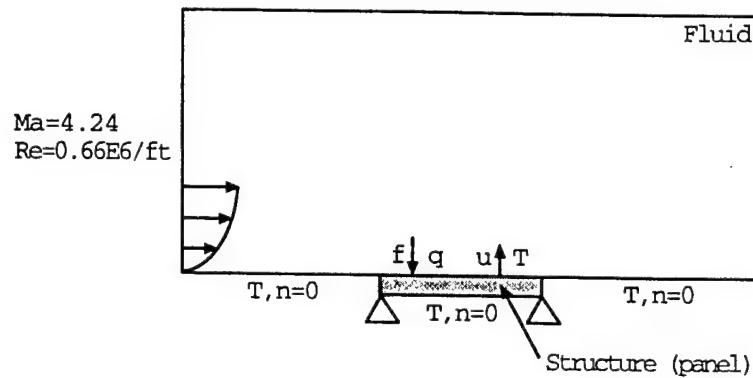


Figure 8 Panel: Boundary Conditions

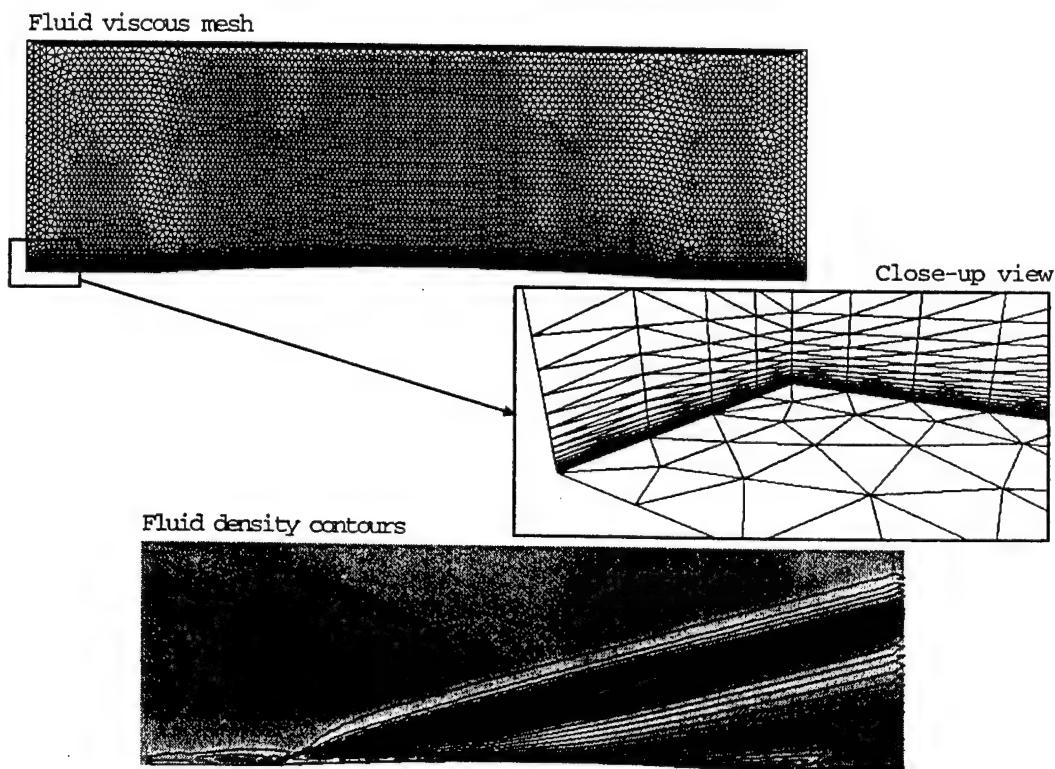


Figure 9 Panel: Grid and Results Obtained

## APPENDIX 5: BREAKUP AND TOPOLOGY CHANGE

# FLUID-STRUCTURE INTERACTION ALGORITHMS FOR RUPTURE AND TOPOLOGY CHANGE

Rainald Löhner<sup>1</sup>, Chi Yang<sup>1</sup>, Juan Cebral<sup>1</sup>, Joseph D. Baum<sup>2</sup>  
Hong Luo<sup>2</sup>, Eric Mestreau<sup>2</sup>, Daniele Pelessone<sup>3</sup> and Charles Charman<sup>3</sup>

<sup>1</sup>Institute for Computational Sciences and Informatics  
MS 4C7, George Mason University, Fairfax, VA 22030, USA  
<http://www.science.gmu.edu/~rlöhner>

<sup>2</sup>Center for Hydrodynamics, Advanced Technology Group  
Science Applications International Corporation  
1710 Goodridge Drive, MS 2-3-1, McLean, VA 22102, USA

<sup>3</sup>General Atomics, San Diego, CA 92121, USA

**Abstract.** Essential methodologies for the routine simulation of fluid-structure interaction problems where rupture and topology change are present are developed. These include: Automatic surface and topology reconstruction, automatic grid sizing techniques, parallel remeshing and interpolation algorithms for topologically different domains. Numerical examples show the effectiveness of the developed methodology.

**Keywords.** Fluid-Structure Interaction, CFD, CSD, Topology Change

## 1. INTRODUCTION

The advent of advanced numerical techniques, affordable 3-D graphics and powerful compute servers over the last decade has led to a rapid maturing of the 'core' disciplines that encompass Computational Mechanics: Computational Structural Dynamics (CSD), Computational Fluid Dynamics (CFD) and Computational Thermodynamics (CTD). The next logical step has been the development of algorithms for the solution of interdisciplinary problems. An important class of problems that fall under this category is given by Fluid-Structure interaction. These problems are characterized by changes of (structural) geometry due to fluid pressure, shear and heat loads that have a considerable effect on the flowfield, changing the loads in turn. Examples of industrial problems that fall under this category are: steady-state aerodynamics of wings under cruise conditions, such as civilian and military planes; aeroelasticity of vibrating, i.e. elastic structures, such as flutter [Bat88, Bos93, Byu94, Rau93, Alo95, Byu98, Kim99] and buzz (aeroplanes, nozzles, turbines), galloping (cables, bridges [Kva99]), noise control [Eve90, Eve91] (cars, trains, submarines) and maneuvering and control (missiles, drones); 'weak and nonlinear' structures, such as wetted

membranes (parachutes, airbags [Mes96], parasols, tents, sails [Jac87]) and biological tissues (hearts, lungs); 'strong and nonlinear' structures, such as shock-structure interaction (command and control centers, bunkers, vehicles [Bau93, Bau96, Kam96], weapon fragmentation [Bau98, Bau99]), hypersonic flight vehicles [Tho88, Löh98b]; and variable geometry vehicles.

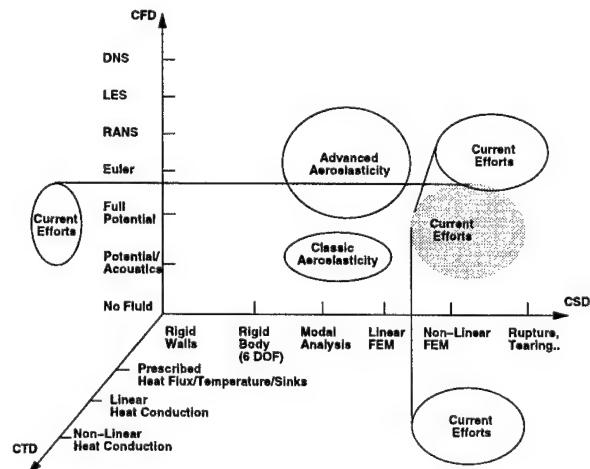


Figure 1 Fluid-Structure-Thermal Interaction

The most important question is how to combine these different disciplines to arrive at an accurate, modular and cost-effective simulation approach that can handle an arbitrary number of disciplines at the same time. Considering the fluid-structure-thermal interaction problem as an example, we see from the list of possibilities displayed in Figure 1 that any multi-disciplinary capability must have the ability to quickly switch between approximation levels, models, and ultimately codes. It is clear that only those approaches that allow maximum flexibility, i.e.:

- Linear and nonlinear CFD, CSD and CTD models;
- Different, optimally suited discretizations for CFD, CSD and CTD domains;
- Modularity in CFD, CSD and CTD models and codes;
- Fast multi-disciplinary problem definition;
- Fully automatic grid generation for arbitrary geometrical complexity; and
- Integrated multi-disciplinary visualization [Ceb98] and data reduction

will survive in the long run. The question of how to couple CSD and CFD codes has been treated extensively in the literature. Two main approaches have been pursued to date: strong coupling and loose coupling. The strong or tight coupling technique solves the discrete system of coupled, nonlinear equations resulting from the CFD, CSD, CTD and interface conditions **in a single step**. For an extreme example of the tight coupling approach, where even the discretization on the surfaces was forced to be the same, see [Tho88]. The loose coupling technique solves the same system using an iterative strategy of repeated 'CFD solution followed by CTD solution followed by CSD solution' until convergence is achieved (see Figure 2).

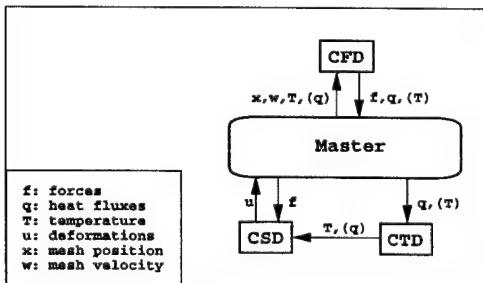


Figure 2 Loose Coupling

Special cases of this second approach include the direct coupling in time of explicit CFD and CSD codes and the incremental load approach of steady aero- and hydro-elasticity. The variables on the boundaries are transferred back and forth between the different codes by a master code that directs the multi-disciplinary run. Each code (CFD, CSD, CTD, ..) is seen as a subroutine, or object, that is called by the master code, or as a series of processes that communicate via message passing. This implies that the transfer of geometrical and physical information is performed between the different codes without affecting their efficiency, layout, basic functionality, and coding styles. At the same time, different CSD, CTD or CFD codes may be replaced, making this a very modular approach. This allows for a straightforward re-use of existing codes and the choice of the 'best model' for a given application. The information transfer software may be developed, to a large extent, independently from the CSD, CTD and CFD codes involved, again leading to modularity and software reuse. For this reason, this approach has gained widespread acceptance [Gur90, Löh90, Rau93, Gur93, Byu94, Löh95a, Bau96, Mes96, Coc97, Bau98, Byu98, Gri98, Löh98b, Bau99, Kim99].

Optimal discretizations for the CSD and CFD problem are, in all probability, not going to be the same. As an example, consider a commercial aircraft wing undergoing aeroelastic loads. For an accurate CFD solution using the Euler equations, an accurate surface representation with 60-120 points in the chord-direction will be required. For the CSD model, a  $20 \times 40$  mesh of plate elements may be more than sufficient. Any general fluid-structure coupling strategy must be able to handle efficiently the information transfer between different surface representations. This is not only a matter of fast interpolation techniques [Löh95b, Mam95], but also of accuracy, load conservation [Ceb97a,b], geometrical fidelity [Ceb97b], and temporal synchronization [Les96, Ceb97b]. When considering different mesh sizes for the CSD and CFD surface representation, the enforcement of accuracy in the sense of:

$$\sigma_s(\mathbf{x}) \approx \sigma_f(\mathbf{x}) \quad (1)$$

and conservation in the sense of:

$$\mathbf{f} = \int \sigma_s \mathbf{n} d\Gamma = \int \sigma_f \mathbf{n} d\Gamma \quad (2)$$

proves to be non-trivial. The best way to date to handle this problem for similar surface representations is via an adaptive Gaussian quadrature [Ceb97a]. In many instances, the CSD model will either be coarse as compared to the CFD model, or may even be on a different modeling or abstraction level. On the other hand, it is the CSD model that dictates the deformation of the CFD surface. It is not difficult to see that an improper transfer of CSD deformation to the CFD surface mesh can quickly lead to loss of geometrical fidelity. Although a number of recovery techniques have been proposed to date [Gur93, Löh96, Ceb97b], the proper surface deformation of ‘non-glued’ CFD, CSD and CTD surfaces, and an error indicator to warn the unsuspecting user, still represent unanswered questions.

Having described the broader context of fluid-structure-thermal interaction, we now turn our attention to a particular class of problems: shock-structure interaction, with rupture and topology change. After reviewing the possible approaches in Section 2, the elements required for a satisfactory solution of this class of problems are dealt with in turn: Surface Reconstruction and Definition (Section 3), Automatic Remeshing (Sections 4,5) and Interpolation (Section 6). Some examples are given in Section 7. Finally, in Section 8, conclusions are drawn and future work areas are described.

## 2. SHOCK-STRUCTURE INTERACTION

The interaction of strong shocks with structures will, in most instances, lead to some form of structural failure (cracking, spalation, breach, failure, collapse, etc.). Within a loose coupling approach (see Figure 2), this implies that the ‘wetted CSD surface’ will change in time. This change in structural integrity also leads to new zones or volumes for the fluid that formerly were not present, implying that automatic gridding based on discrete surface data is essential. Within CFD, two grid systems have appeared that can handle automatically such time- and topology-varying geometries:

- a) Rigid, non-surface-conforming grids and
- b) Moving, surface-conforming grids.

We briefly describe the advantages and disadvantages of both classes.

### a) Rigid Grids:

Here, the grid is basically laid over the volume to be gridded. Adaptivity based on surface curvature

or geometric stiffness is used to refine the grid close to bodies immersed in the flowfield. The cells cut by surfaces are identified and treated with special procedures. The advantages of this approach are manifold:

- Construction of grids is extremely fast;
- The grid is not moving, leading to a faster solver;
- Most of the algorithmic work is concentrated in identifying and treating cut surface cells;

However, the approach also has disadvantages:

- A considerable portion of the grid may be ‘blacked out’;
- New cut cells and boundary conditions have to be established every timestep;
- Topological consistency (water-tightness) after cutting cells is not always easy to establish;
- The cut cells may become very small (the treatment of small cells has become the focus of much research [Ber92, Pem95, Aft97, Lan97]);
- Grids suitable for RANS calculations are difficult to construct.

### b) Moving Grids:

This approach starts with a surface-conforming grid. As the structure deforms, the mesh is moved. Excessively distorted or negative elements are automatically removed. The ensuing voids are remeshed and the solution interpolated. If topological changes occur, the surface and volume mesh have to be regenerated, and the solution reinterpolated. This approach would appear to have a lot of disadvantages:

- Moving grid (ALE) solvers are more expensive than fixed grid solvers (recalculation of Jacobians, extra fluxes, etc.);
- Geometric conservation laws have to be obeyed, placing constraints on the solver [Les96];
- Moving the mesh requires extra computing resources;
- Fast remeshing and interpolation algorithms are required.

However, the approach also has advantages:

- Topological consistency (water-tightness) is easy to establish;
- It is possible to construct grids suitable for RANS calculations.

Over the last decade, we have pursued the second approach, attempting to remedy the disadvantages by:

- Fast ALE edge-based solvers [Luo93, Luo94, Bau94, Bau95, Löh98a];

- Mass-matrix-based satisfaction of geometric conservation laws [Bau95];
- Optimal mesh movement algorithms [Löh96b];
- Local remeshing [Löh90, Bau98];
- Parallel unstructured grid generation [Löh96a, Löh99] and
- Optimal interpolation techniques for unstructured grids [Löh95b].

In the sequel, we describe recent developments that have targeted the class of problems considered here: fluid-structure interaction with rupture and topology change.

### 3. SURFACE RECONSTRUCTION

Suppose that due to cracking, failure, spallation, etc., the ‘wetted surface’ of the CSD domain has been changed. This new surface, given by a list of points and faces, has to be matched with a corresponding CFD surface. The CFD surface data typically consists of surface segments defined by analytical functions that do not change in time (such as exterior walls, farfield boundaries, etc.), and surface segments defined by triangulations (i.e. discrete data) that change in time. These triangulations are obtained from the ‘wetted CSD surface’ at every timestep. When a change in topology is detected, the new surface definition is recovered from the discrete data, and joined to the surfaces defined analytically, as indicated in Figure 3.

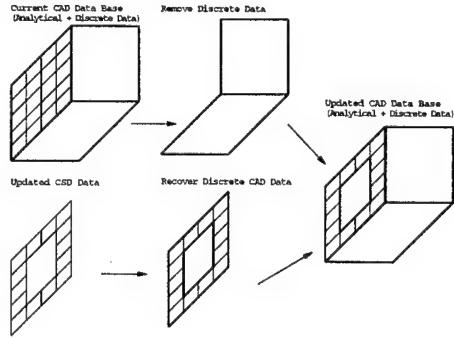


Figure 3 Automatic Surface Reconstruction

The discrete surface is defined by a support triangulation, with lines and end-points to delimit its boundaries. In this sense, the only difference with analytically defined surfaces is the (discrete) support triangulation. The patches, lines and end-points of the ‘wetted CSD surface’ are identified by comparing the unit surface normals of adjacent faces. If

the scalar product of them lies below a certain tolerance, a ridge is defined. **Corners** are defined as points that are attached to:

- Only one ridge;
- More than two ridges; or
- Two ridges with considerable deviation of unit side-vector.

Between corners, the ridges form **discrete lines**. These discrete lines either separate or are embedded completely (i.e. used twice) in **discrete surface patches**. Figure 4 sketches the recovery of surface features and the definition of discrete surface patches for a simple configuration. For more information, see [Löh96c].

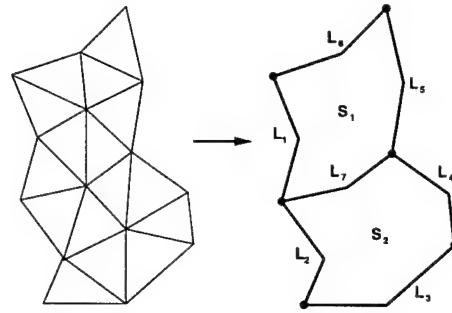


Figure 4 Discrete Surface Recovery

For the old surface definition data set, the surface patches attached to wetted CSD surfaces are identified and all information associated with them is discarded. The remaining data is then joined to the new wetted CSD surface data, producing the updated surface definition data set. This data set is then used to generate the new surface and volume grids.

The surface reconstruction procedure may be summarized as follows:

- For the Updated Discrete Data, Obtain:
  - Surface Patches + B.C.
  - Lines
  - End-Points
  - Sources
- For the Old Analytical+Discrete Data:
  - Remove Discrete Data
  - Reorder Arrays
- Merge:
  - Old Analytical Data
  - Updated Discrete Data

#### 4. ELEMENT SIZE AND SHAPE

Cracking, failure, spalation, etc. will lead to the appearance of many fragments or chunks in the flowfield. The objects that are flying in the flowfield come in many different sizes, and yet require sufficient surface and volume resolution in order for the CFD solver to yield acceptable forces, moments and ultimately trajectories. A very simple way to obtain surface grids that are acceptable is by defining a maximum element size that is linked to the size of the flying object. This approach can lead to highly distorted volume grids when neighbouring objects with very different surface discretizations are present. Compatible surface and volume grids may be obtained by using either adaptive background grids or sources [Löh96a, Löh97]. In the present case, point sources were employed. The element size as a function of the non-dimensional distance from the source  $\xi(\mathbf{x})$ :

$$\delta(\mathbf{x}) = \delta(\xi(\mathbf{x})) ; \xi = \max \left( 0, \frac{r(\mathbf{x}) - r_0}{r_s} \right) , \quad (3)$$

where  $r(\mathbf{x})$  is the shortest distance from the source to  $\mathbf{x}$ ,  $r_0$  the so-called zone of constant element size, and  $r_s$  the scaling length, is given by the polynomial:

$$\delta(\xi) = a_0 + a_1 \xi + a_2 \xi^2 . \quad (4)$$

Once the individual fragments or chunks are identified using an advancing front neighbour search over the discrete surface patches, the center of gravity and the average radius can be computed. This leads to the desired mesh size parameter  $a_0$ . The parameters  $a_1, a_2$  are then obtained from mesh smoothness considerations.

#### 5. AUTOMATIC PARALLEL REMESHING

Over the last five years, major efforts have been devoted to harness the power of parallel computer platforms. While many CFD and CSD solvers have been ported to parallel machines, grid generators have lagged behind. For applications where remeshing is an integral part of simulations, e.g. problems with moving bodies [Löh90, Mes93, Mes95, Bau96, Kam96, Löh98b, Has98] or changing topologies [Bau98, Bau99], the time required for mesh regeneration can easily consume more than 50% of the total time required to solve the problem. Faced with this situation, a number of efforts have been reported

on parallel grid generation [Löh92, dCo94, Sho95, dCo95, Oku96, Che97, Oku97, Sai99].

The two most common ways of generating unstructured grids are the Advancing Front Technique (AFT) [Per87, Per88, Löh88a,b, Per90, Per92, Jin93, Fry94, Löh96] and the Generalized Delaunay Triangulation (GDT) [Bak89, Geo91, Wea92, Wea94, Mar95]. The AFT introduces one element at a time, while the GDT introduces a new point at a time. Thus, both of these techniques are, in principle, scalar by nature, with a large variation in the number of operations required to introduce a new element or point. While coding and data structures may influence the scalar speed of the 'core' AFT or GDT, one often finds that for large-scale applications, the evaluation of the desired element size and shape in space, given by background grids, sources or other means [Löh96] consumes the largest fraction of the total grid generation time. Unstructured grid generators based on the AFT may be parallelized by invoking distance arguments, i.e. the introduction of a new element only affects (and is affected by) the immediate vicinity. This allows for the introduction of elements in parallel, provided that sufficient distance lies between them.

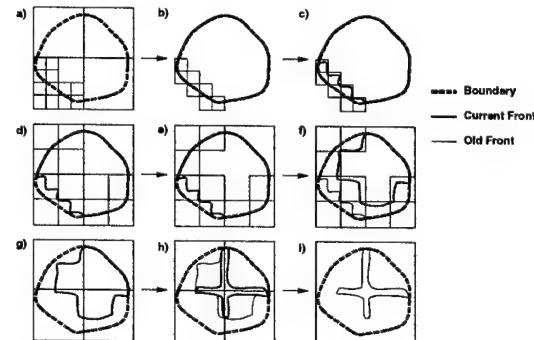


Figure 5 Parallel Grid Generation

A convenient way of delimiting the possible zones where elements may be introduced by each processor is via boxes. These boxes may be obtained in a variety of ways, i.e. via bins, binary recursive trees, or octrees. We have found the octree to be the best of these possibilities, particularly for grids with a large variation of element size. In order to recover a parallel gridding procedure that resembles closely the advancing front technique on scalar machines, only the boxes covering the active front in regions where the smallest new elements are being introduced are considered. This has been shown schematically in Fig-

ure 5a,b for a simle 2-D domain. After these boxes have been filled with elements (Figure 5c), the process starts anew: a new octree is built (Figure 5d), new boxes are created (Figure 5e) and meshed in parallel (Figure 5f). This cycle is repeated until no faces are left in the active front (Figures 5g-i).

At the end of each parallel gridding pass, each one of the boxes gridded can have an internal boundary of faces. For a large number of boxes, this could result in a very large number of faces for the active front. This problem can be avoided by shifting the boxes slightly, and then regridding them again in parallel, as shown in Figure 6. This simple technique has the effect of eliminating almost all of the faces between boxes with a minor modification of the basic parallel gridding algorithm.

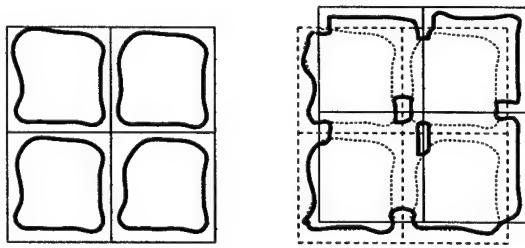


Figure 6 Shift and Regrid Technique

If we define as  $d_{min}$  the minimum element size in the active front, and as  $s_{min}$  the minimum box size in which elements are to be generated, the parallel AFT proceeds as follows:

**WHILE:** There are active faces left:

- Form an octree with minimum octant size  $s_{min}$  for the active points;
- Retain the octants that have faces that will generate elements of size  $d_{min}$  to  $c_l \cdot d_{min}$ ;
- If too many octants are left: agglomerate them into boxes;
- **DO ISHFT=0,2:**
  - **IF:** ISHFT.NE.0:
    - Shift the boxes by a preset amount;
  - **ENDIF**
  - Generate, in parallel, elements in these boxes, allowing only elements up to a size of  $c_l \cdot d_{min}$ ;
- **ENDDO**
- Increase  $d_{min} = 1.5 * d_{min}$ ,  $s_{min} = 1.5 * s_{min}$ ;

**ENDWHILE**

The increase factor allowed is typically in the range  $c_l = 1.5 - 2.0$ . Figures 7a-c show an example obtained on an SGI Origin 2000 running in shared

memory mode. The case considered is the garage of an office complex, and had approximately 9.2 million tetrahedra. As one can see, although not perfect, speedups are comparable to those of production CFD codes. For more details on the parallel grid generator, see [Löh99]

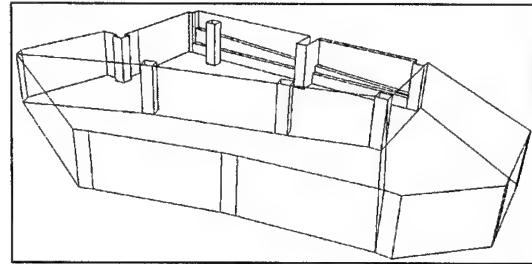


Figure 7a Garage: Wireframe

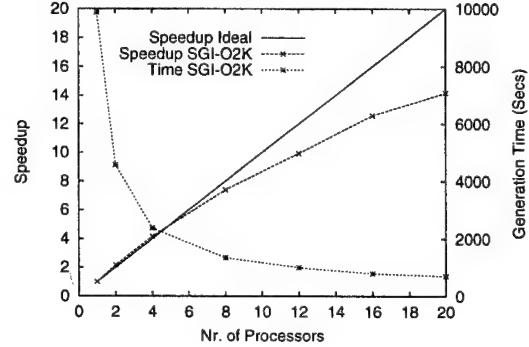


Figure 7b Garage: Speedups Obtained

## 6. INTERPOLATION WITH TOPOLOGY CHANGE

Once a new mesh has been generated, the solution from the previous timestep (on the previous mesh) has to be interpolated. A series of optimal interpolation algorithms for unstructured grids have been described in [Löh95b]. Whenever new fluid domains are created due to failure, cracking and spalation, interpolating the fluid solution from the previous timestep to these new domains will end in failure, as there are no possible host elements in the old mesh. Figure 8 shows a typical case where a wall that initially separates two rooms breaks, changing the topology of the fluid and solid domains.

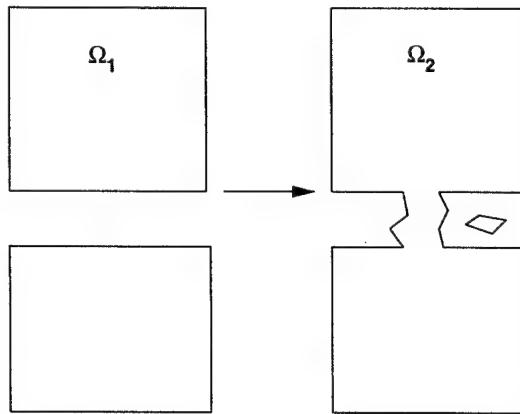


Figure 8 Interpolation With Different Domains

It is therefore important to identify points of the new mesh that lie outside the confines of the old mesh. A simple way that has proven successful is to form a Cartesian mesh or bins (Figure 9). A loop is then performed over the elements of the old mesh, marking the bins covered by elements. In a second loop over the points of the new grid, all points that fall into bins not covered by the old grid are marked as impossible to interpolate. This procedure can be done recursively by obtaining the confines of the volume where points have been marked as impossible, leading to so-called 'telescoping' of the bin search region.

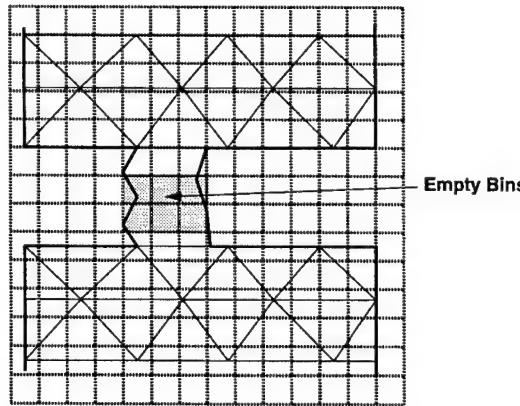


Figure 9 Marking Impossible Points

No attempt is then made to interpolate the points marked as outside the old mesh. The unknowns for

these points can be extrapolated using different procedures:

- Advancing layers (most often used for sub-sonic/isotropic flows);
- Upstream (used primarily for supersonic flows);
- Closest known point (for cracks); or
- Via user-prescribed subroutine (for special cases).

## 7. EXAMPLES

Two examples are included that show the effectiveness of the procedures developed to date. In both instances, the CSD solver used is GA-DYNA [Pel95, Pel97, Pel98] and the CFD solver used is FEM-FCT within FEFLO98 [Löh87, ].

### 6.1 Fragmenting Weapon:

The first case considered is that of a fragmenting weapon. The detonation and shock propagation were modeled using a JWL equation of state. At the beginning, the walls of the weapon separate two flow domains: the inner one, consisting of high explosive, and the outer one, consisting of air. As the structure of the weapon begins to fail, fragments are shrunk and the ensuing gaps are automatically remeshed, leading to one continuous domain. The mesh in the fluid domain was adapted using sources for geometric fidelity [Löh96a] and the modified H2-seminorm error indicator proposed in [Löh87, Löh92]. At the end of the run, the flow domain contains approximately 750 independently flying bodies and 16 million elements. Figures 8-10 show the development of the detonation. The fragmentation of the weapon is clearly visible. Figure 11 shows the correlation with the observed experimental evidence.

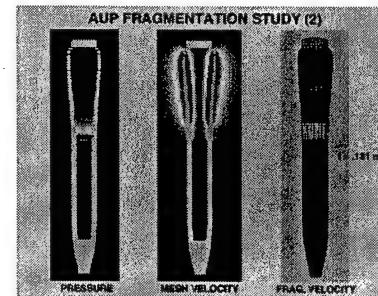


Figure 10a Fragmenting weapon at 131msec

Euler equations with ideal gas equation of state.

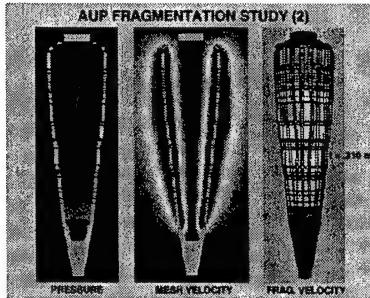


Figure 10b Fragmenting weapon at 310msec

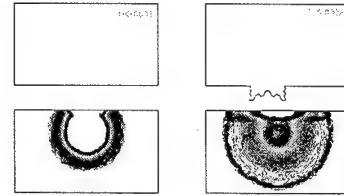


Figure 11a Pressure at  $t=0.0001, 0.0002$

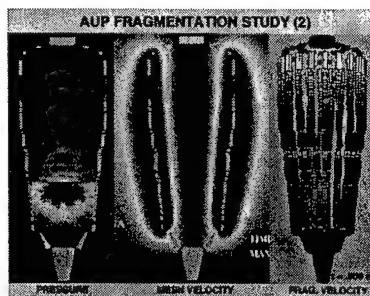


Figure 10c Fragmenting weapon at 500msec

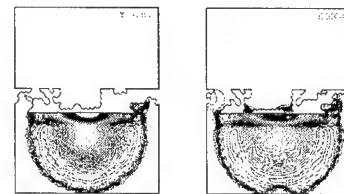


Figure 11b Pressure at  $t=0.0003, 0.0004$

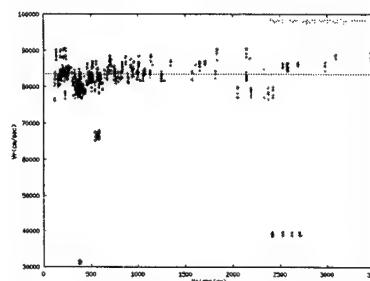


Figure 10d Radial velocity as a function of fragment weight

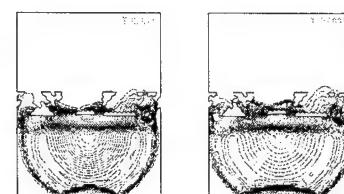


Figure 11c Pressure at  $t=0.0005, 0.0006$

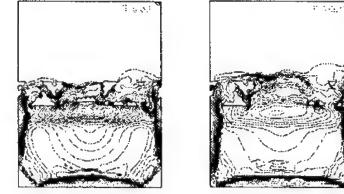


Figure 11d Pressure at  $t=0.0008, 0.0010$

## 6.2 Blast Close to a Wall

This example shows the capabilities of the present fluid-structure interaction methodology to treat situations with extreme transient loading, plastic deformation, failure, breakup, topology change and automatic remeshing. The blast was modeled using the

Initially, a 'high energy' state given by:  $\rho =$

$0.25 \text{ gr/cm}^3, v = 0.00 \text{ cm/sec}, p = 0.43e + 09 \text{ dynes}$  within the sphere with radius  $r = 8.0 \text{ cm}$  centered at  $x = 34.29 \text{ cm}, y = -10.00 \text{ cm}, z = 19.05 \text{ cm}$

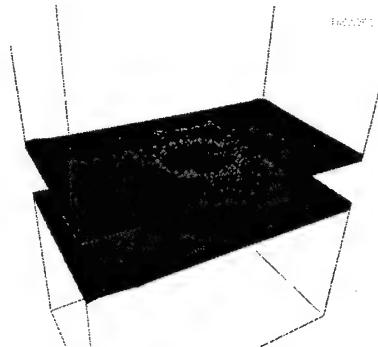


Figure 12a Surface Velocity at  $t=0.0002$

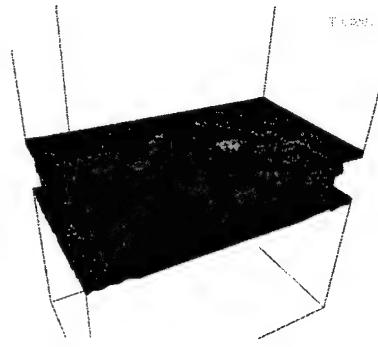


Figure 12b Surface Velocity at  $t=0.0004$

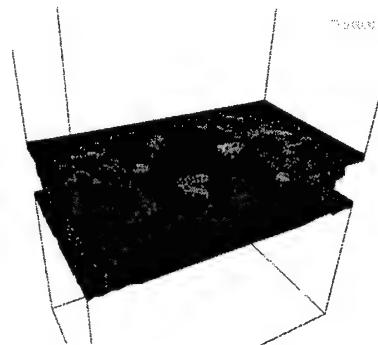


Figure 12c Surface Velocity at  $t=0.0006$

in the lower room, and a quiescent air state given

by:  $\rho = 0.00104 \text{ gr/cm}^3, v = 0.00 \text{ cm/sec}, p = 0.86185e + 06 \text{ dynes}$  was prescribed. The wall was assumed to be of reinforced concrete. The material model used was elasto/perfect plastic. Failure was assumed to occur when the average strain in an element exceeded 60%. At the beginning, the wall separates the two rooms. As the wall begins to fail the larger fragments are shrunk while keeping their mass and moments of inertia intact, leading to gaps. The smaller fragments are removed completely from the structure and converted to spheres that exchange mass, momentum and energy with the fluid, exchange momentum with the remaining structure via contact, but can not interact between them, nor 'occupy space'. The ensuing gaps that result due to shrinkage or element removal are automatically remeshed and filled with fluid. At some point during the run, the flow domain contains approximately 250 independently flying bodies and 1 million elements. Figures 11a-d show a planar cut through the two rooms. The impact of the blast wave on the wall, its reflection, the ensuing wall failure and eventual penetration of the blast wave into the adjacent room are clearly visible. Figures 12a-c show the surface velocity and fragments of the wall.

## 8. CONCLUSIONS AND OUTLOOK

Several methodologies that are essential for the routine simulation of fluid-structure interaction problems where rupture and topology change are present have been developed. These include:

- Surface and topology reconstruction;
- Automatic sizing techniques;
- Parallel remeshing; and
- Interpolation algorithms for topologically different domains.

Numerical examples indicate that with the developed methodology a new level of realism and sophistication has been achieved for this class of problems. Among the many outstanding issues we just mention:

- Parallelization for distributed memory machines;
- Automatic remeshing for RANS simulations; and
- Proper treatment of cracking.

At the threshold of a new century, we envision a multi-disciplinary, database-linked framework that is accessible from anywhere on demand, simulations with unprecedented detail and realism carried out

in fast succession, virtual meeting spaces where geographically displaced designers and engineers discuss and analyze collaboratively new ideas, and first-principles driven virtual reality.

## 9. ACKNOWLEDGEMENTS

This research was partially supported by AFOSR and DTRA. Drs. Leonidas Sakell, Michael Giltrud and Darren Rice acted as technical monitors.

## 10. REFERENCES

- [Aft97] M.J. Aftosmis, M.J. Berger and J.E. Melton - Robust and Efficient Cartesian Mesh Generation for Component-Based Geometry; *AIAA-97-0196* (1997).
- [Alo95] J. Alonso, L. Martinelli and A. Jameson - Multigrid Unsteady Navier-Stokes Calculations with Aeroelastic Applications; *AIAA-95-0048* (1995).
- [Bak89] T.J. Baker - Developments and Trends in Three-Dimensional Mesh Generation. *Appl. Num. Math.* **5**, 275-304 (1989).
- [Bat88] J.T. Batina, R.M. Bennet, D.A. Seidel, H.J. Cunningham and S.R. Bland - Recent Advances in Transonic Computational Aeroelasticity; *Comp. Struct.* **30**, No.1/2, 29-37, (1988).
- [Bau93] J.D. Baum, H. Luo and R. Löhner - Numerical Simulation of a Blast Inside a Boeing 747; *AIAA-93-3091* (1993).
- [Bau94] J.D. Baum, H. Luo and R. Löhner - A New ALE Adaptive Unstructured Methodology for the Simulation of Moving Bodies; *AIAA-94-0414* (1994).
- [Bau95] J.D. Baum, H. Luo and R. Löhner - Validation of a New ALE, Adaptive Unstructured Moving Body Methodology for Multi-Store Ejection Simulations; *AIAA-95-1792* (1995).
- [Bau96] J.D. Baum, H. Luo, R. Löhner, C. Yang, D. Pelessone and C. Charman - A Coupled Fluid/Structure Modeling of Shock Interaction with a Truck; *AIAA-96-0795* (1996).
- [Bau98] J.D. Baum, H. Luo and R. Löhner - The Numerical Simulation of Strongly Unsteady Flows With Hundreds of Moving Bodies; *AIAA-98-0788* (1998).
- [Bau99] J.D. Baum, H. Luo, E. Mestreau, R. Löhner, D. Pelessone and C. Charman - A Coupled CFD/CSD Methodology for Modeling Weapon Detonation and Fragmentation; *AIAA-99-0794* (1999).
- [Ber92] M.J. Berger and R.J. LeVeque - An Adaptive Cartesian Mesh Algorithm for the Euler Equations in Arbitrary Geometries; *AIAA-92-0443* (1992).
- [Bos93] A.H. Boschitsch and T.R. Quackenbush - High Accuracy Computations of Fluid-Structure Interaction in Transonic Cascades; *AIAA-93-0485* (1993).
- [Byu94] C. Byun and G.P. Guruswamy - Wing-Body Aeroelasticity Using Finite-Difference Fluid/Finite-Element Structural Equations on Parallel Computers; *AIAA-94-1487* (1994).
- [Byu98] C. Byun and G.P. Guruswamy - Aeroelastic Computations on Wing- Body- Control Configurations on Parallel Computers; *J. Aircraft* **35**, 288-294 (1998).
- [Ceb97a] J.R. Cebral and R. Löhner - Conservative Load Projection and Tracking for Fluid-Structure Problems; *AIAA J.* **35**, 4, 687-692 (1997).
- [Ceb97b] J.R. Cebral and R. Löhner - Fluid-Structure Coupling: Extensions and Improvements; *AIAA-97-0858* (1997).
- [Ceb98] J.R. Cebral and R. Löhner - Interactive On-Line Visualization and Collaboration for Parallel Unstructured Multidisciplinary Applications; *AIAA-98-0077* (1998).
- [Che97] L.P. Chew, N. Chrisochoides and F. Sukup - Parallel Constrained Delaunay Meshing; *Proc. 1997 Workshop on Trends in Unstructured Mesh Generation*, June (1997).
- [Coc97] COCOLIB Deliverable 1.1: Specification of the COupling COmmunications LIBrary; CISPAR ESPRIT Project 20161, See <http://www.pallas.de/cispar/pages/docu.htm> (1997).
- [dCo94] H.L. de Cougny, M.S. Shephard and C. Ozturan - Parallel Three-Dimensional Mesh Generation; *Computing Systems in Engineering* **5**, 311-323 (1994).
- [dCo95] H.L. de Cougny, M.S. Shephard and C. Ozturan - Parallel Three-Dimensional Mesh Generation on Distributed Memory MIMD Computers; *Tech. Rep. SCOREC Rep. # 7*, Rensselaer Polytechnic Institute (1995).
- [Don82] J. Donea - An Arbitrary Lagrangian-Eulerian Finite Element Method for Transient Dynamic Fluid-Structure Interactions; *Comp. Meth. Appl. Mech. Eng.* **33**, 689-723 (1982).
- [Eve90] G.C. Everstine and F.M. Henderson - Coupled Finite Element/Boundary Element Approach

for Fluid-Structure Interaction; *J. Acoust. Soc. Am.* 87, 5, 1938-1947 (1990).

[Eve91] G.C. Everstine - Prediction of Low Frequency Vibrational Frequencies of Submerged Structures; *J. Vibrations and Acoustics* 113, (1991).

[Fel93] F.F. Felker - Direct Solution of Two-Dimensional Navier-Stokes Equations for Static Aeroelasticity Problems; *AIAA J.* 31, 1, 148-153 (1993).

[Fry94] J. Frykestig - Advancing Front Mesh Generation Techniques with Application to the Finite Element Method; *Pub. 94:10*, Chalmers University of Technology; Göteborg, Sweden (1994).

[Geo91] P.L. George, F. Hecht and E. Saltel - Automatic Mesh Generator With Specified Boundary; *Comp. Meth. Appl. Mech. Eng.* 92, 269-288 (1991).

[Gur90] G.P. Guruswamy - Unsteady Aerodynamic and Aerelastic Calculations for Wings Using Euler Equations; *AIAA J.* 28, 3, 461-469 (1990).

[Gur93] G.P. Guruswamy and C. Byun - Fluid-Structural Interactions Using Navier-Stokes Flow Equations Coupled with Shell Finite Element Structures; *AIAA-93-3087* (1993).

[Has98] O. Hassan, L.B. Bayne, K. Morgan and N. P. Weatherill - An Adaptive Unstructured Mesh Method for Transient Flows Involving Moving Boundaries; pp. 662-674 in *Computational Fluid Dynamics '98* (K.D. Papailiou, D. Tsahalis, J. Périoux and D. Knörzer eds. ) Wiley (1998).

[Jac87] P.S. Jackson and G.W. Christie - Numerical Analysis of Three-Dimensional Elastic Membrane Wings; *AIAA J.* 25, 5, 676-682, (1987).

[Jin93] H. Jin and R.I. Tanner - Generation of Unstructured Tetrahedral Meshes by the Advancing Front Technique; *Int. J. Num. Meth. Eng.* 36, 1805-1823 (1993).

[Kam96] A. Kamoulakos, V. Chen, E. Mestreau and R. Löhner - Finite Element Modelling of Fluid/Structure Interaction in Explosively Loaded Aircraft Fuselage Panels Using PAMSHOCK/PAMFLOW Coupling; *Conf. on Spacecraft Structures, Materials and Mechanical Testing*, Noordwijk, The Netherlands, March (1996).

[Kim99] T. Kimura, H. Takemiyia and R. Onishi - CFD/CSD Coupled Simulation on Parallel Computer Cluster; *AIAA-99-3275-CP* (1999).

[Kva99] T. Kvamsdal et al. eds. - *Computational Methods for Fluid-Structure Interaction*, Tapir Press (1999).

[Lan97] A.M. Landsberg and J.P. Boris - The Virtual Cell Embedding Method: A Simple Approach for Gridding Complex Geometries; *AIAA-97-1982* (1997).

[Les96] M. Lesoinne and Ch. Farhat - Geometric Conservation Laws for Flow Problems With Moving Boundaries and Deformable Meshes, and Their Impact on Aeroelastic Computations; *Comp. Meth. Appl. Mech. Eng.* 134, 71-90 (1996).

[Löh87] R. Löhner, K. Morgan, J. Peraire and M. Vahdati - Finite Element Flux-Corrected Transport (FEM-FCT) for the Euler and Navier-Stokes Equations; *Int. J. Num. Meth. Fluids* 7, 1093-1109 (1987).

[Löh88a] R. Löhner - Some Useful Data Structures for the Generation of Unstructured Grids; *Comm. Appl. Num. Meth.* 4, 123-135 (1988).

[Löh88b] R. Löhner and P. Parikh - Three-Dimensional Grid Generation by the Advancing Front Method; *Int. J. Num. Meth. Fluids* 8, 1135-1149 (1988).

[Löh90] R. Löhner - Three-Dimensional Fluid-Structure Interaction Using a Finite Element Solver and Adaptive Remeshing; *Computer Systems in Engineering* 1, 2-4, 257-272 (1990).

[Löh92] R. Löhner and J.D. Baum - Adaptive H-Refinement on 3-D Unstructured Grids for Transient Problems; *Int. J. Num. Meth. Fluids* 14, 1407-1419 (1992).

[Löh95a] R. Löhner, C. Yang, J. Cebral, J.D. Baum, H. Luo, D. Pelessone and C. Charman - Fluid-Structure Interaction Using a Loose Coupling Algorithm and Adaptive Unstructured Grids; *AIAA-95-2259* [Invited] (1995).

[Löh95b] R. Löhner - Robust, Vectorized Search Algorithms for Interpolation on Unstructured Grids; *J. Comp. Phys.* 118, 380-387 (1995).

[Löh95c] R. Löhner - Mesh Adaptation in Fluid Mechanics; *Eng. Fracture Mech.* 50, 819-847 (1995).

[Löh96a] R. Löhner - Extending the Range of Applicability and Automation of the Advancing Front Grid Generation Technique; *AIAA-96-0033* (1996).

[Löh96b] R. Löhner and Chi Yang - Improved ALE Mesh Velocities for Moving Bodies; *Comm. Num. Meth. Eng.* 12, 599-608 (1996).

[Löh96c] R. Löhner - Regridding Surface Triangulations; *J. Comp. Phys.* 126, 1-10 (1996).

[Löh97] R. Löhner - Automatic Unstructured Grid Generators; *Finite Elements in Analysis and Design*

25, 111-134 (1997).

[Löh98a] R. Löhner - Renumbering Strategies for Unstructured- Grid Solvers Operating on Shared- Memory, Cache- Based Parallel Machines; *Comp. Meth. Appl. Mech. Eng.* 163, 95-109 (1998).

[Löh98b] R. Löhner, C. Yang, J. Cebral, J.D. Baum, H. Luo, D. Pelessone and C. Charman - Fluid-Structure-Thermal Interaction Using a Loose Coupling Algorithm and Adaptive Unstructured Grids; *AIAA-98-2419 [Invited]* (1998).

[Löh99] R. Löhner and J. Cebral - Parallel Advancing Front Grid Generation; *Proc. 8th Int. Meshing Roundtable*, October (1999) <http://www.cfd.sandia.gov/8imr.html>.

[Luo93] H. Luo, J.D. Baum, R. Löhner and J. Cabello - Adaptive Edge-Based Finite Element Schemes for the Euler and Navier-Stokes Equations; *AIAA-93-0336* (1993).

[Luo94] H. Luo, J.D. Baum and R. Löhner - Edge-Based Finite Element Scheme for the Euler Equations; *AIAA J.* 32, 6, 1183-1190 (1994).

[Mam95] N. Maman and C. Farhat - Matching Fluid and Structure Meshes for Aeroelastic Computations: A Parallel Approach; *Computers and Structures* 54, 4, 779-785 (1995).

[Mar95] D.L. Marcum and N.P. Weatherill - Unstructured Grid Generation Using Iterative Point Insertion and Local Reconnection; *AIAA J.* 33, 9, 1619-1625 (1995).

[Mes93] E. Mestreau, R. Löhner and S. Aita - TGV Tunnel-Entry Simulations Using a Finite Element Code with Automatic Remeshing; *AIAA-93-0890* (1993).

[Mes96] E. Mestreau and R. Löhner - Airbag Simulation Using Fluid/Structure Coupling; *AIAA-96-0798* (1996).

[Oku96] T. Okusanya and J. Peraire - Parallel Unstructured Mesh Generation; *Proc. 5th Int. Conf. Num. Grid Generation in CFD and Related Fields*, Mississippi, April (1996).

[Oku97] T. Okusanya and J. Peraire - 3-D Parallel Unstructured Mesh Generation; *Proc. Joint ASME/ASCE/SES Summer Meeting* (1997).

[Pel95] D. Pelessone and C.M. Charman - Adaptive Finite Element Procedure for Non-Linear Structural Analysis; *1995 ASME/JSME Pressure Vessels and Piping Conference*, Honolulu, Hawaii, July (1995).

[Pel97] D. Pelessone and C.M. Charman - An Adaptive Finite Element Procedure for Structural Analysis of Solids; *1997 ASME Pressure Vessels and Piping Conference*, Orlando, Florida, July (1997).

[Pel98] D. Pelessone and C.M. Charman - A General Formulation of a Contact Algorithm with Node/Face and Edge/Edge Contacts; *1998 ASME Pressure Vessels and Piping Conference*, San Diego, California, July (1998).

[Pem95] R.B. Pember, J.B. Bell, P. Colella, W.Y. Crutchfield and M.L. Welcome - An Adaptive Cartesian Grid Method for Unsteady Compressible Flow in Irregular Regions - *sl J. Comp. Phys.* 120, 278 (1995).

[Per87] J. Peraire, M. Vahdati, K. Morgan and O.C. Zienkiewicz - Adaptive Remeshing for Compressible Flow Computations; *J. Comp. Phys.* 72, 449-466 (1987).

[Per88] J. Peraire, J. Peiro, L. Formaggia K. Morgan and O.C. Zienkiewicz - Finite Element Euler Calculations in Three Dimensions; *Int. J. Num. Meth. Eng.* 26, 2135-2159 (1988).

[Per90] J. Peraire, K. Morgan and J. Peiro - Unstructured Finite Element Mesh Generation and Adaptive Procedures for CFD; *AGARD-CP-464*, 18 (1990).

[Per92] J. Peraire, K. Morgan, and J. Peiro - Adaptive Remeshing in 3-D; *J. Comp. Phys.* (1992).

[Qui94] J.J. Quirk - An Alternative to Unstructured Grids for Computing Gas Dynamic Flows Around Arbitrarily Complex Two-Dimensional Bodies; *Comp. Fluids* 23, 1, 125-142 (1994).

[Rau93] R.D. Rausch, J.T. Batina and H.T.Y. Yang - Three-Dimensional Time-Marching Aerelastic Analyses Using an Unstructured-Grid Euler Method; *AIAA J.* 31, 9, 1626-1633 (1993).

[Tho88] E.A. Thornton and P. Dechaumphai - Coupled Flow, Thermal and Structural Analysis of Aerodynamically Heated Panels; *J. Aircraft* 25, 11, 1052-1059 (1988).

[Wea92] N.P. Weatherill - Delaunay Triangulation in Computational Fluid Dynamics; *Comp. Math. Appl.* 24, 5/6, 129-150 (1992).

[Wea94] N.P. Weatherill and O. Hassan - Efficient Three-Dimensional Delaunay Triangulation with Automatic Point Creation and Imposed Boundary Constraints; *Int. J. Num. Meth. Eng.* 37, 2005-2039 (1994).

## APPENDIX 6: PARALLEL FLOW SOLVERS

Reprinted from

# Computer methods in applied mechanics and engineering

---

Comput. Methods Appl. Mech. Engrg. 163 (1998) 95–109

Renumbering strategies for unstructured-grid solvers operating on  
shared-memory, cache-based parallel machines

Rainald Löhner

*GMU/CSI, The George Mason University, Fairfax, VA 22030-4444, USA*

Received 30 July 1996; revised 18 June 1997



**COMPUTER METHODS IN APPLIED MECHANICS AND ENGINEERING**  
**EDITORS: J.H. ARGYRIS, STUTTGART and LONDON**  
**T.J.R. HUGHES, STANFORD, CA**  
**J.T. ODEN, AUSTIN, TX**

**W. PRAGER**  
**Founding Editor**  
**(deceased 1980)**

**EDITORIAL ADDRESSES**

John H. ARGYRIS  
*Institut für Computer Anwendungen*  
Pfaffenwaldring 27  
D-70569 STUTTGART  
Germany  
(Editorial Office)

Thomas J.R. HUGHES  
*Division of  
Applied Mechanics*  
Durand Building  
Room No. 281  
Stanford University  
STANFORD  
CA 94305-4040, USA

J. Tinsley ODEN  
*The University of Texas*  
*The Texas Institute for  
Computational and  
Applied Mathematics*  
Taylor Hall, 2.400  
AUSTIN  
TX 78712, USA

**ASSOCIATE EDITORS**

K. APPA, *Lake Forest, CA*  
I. BABUŠKA, *Austin, TX*  
A.J. BAKER, *Knoxville, TN*  
T.B. BELYTSCHKO, *Evanston, IL*  
F. BREZZI, *Pavia*  
P.G. CIARLET, *Paris*  
L. DEMKOWICZ, *Austin, TX*  
R.E. EWING, *College Station, TX*  
R. GLOWINSKI, *Houston, TX*

R.W. LEWIS, *Swansea*  
J.L. LIONS, *Paris*  
F.L. LITVIN, *Chicago, IL*  
H. LOMAX, *Moffet Field, CA*  
L.S.D. MORLEY, *Farnborough*  
N. OLHOFF, *Aalborg*  
E. ONATE, *Barcelona*  
M. PAPADRAKAKIS, *Athens*

J. PLANCHARD, *Clamart*  
E. RAMM, *Stuttgart*  
G. STRANG, *Cambridge, MA*  
R.L. TAYLOR, *Berkeley, CA*  
S.Ø. WILLE, *Oslo*  
G. YAGAWA, *Tokyo*  
D. ZHU, *Xi'an*  
O.C. ZIENKIEWICZ, *Swansea*

**ADVISORY EDITORS**

M.P. ARNAL, *Baden*  
J.S. ARORA, *Iowa City, IA*  
K.J. BATHE, *Cambridge, MA*  
P.G. BERGAN, *Høvik*  
J.F. BESELING, *Delft*  
M.O. BRISTEAU, *Le Chesnay*  
C. CANUTO, *Turin*  
J.L. CHENOT, *Valbonne*  
Y.K. CHEUNG, *Hong Kong*  
T.J. CHUNG, *Huntsville, AL*  
T.A. CRUSE, *Nashville, TN*  
E.R. DE ARANTES E OLIVEIRA, *Lisbon*  
J. DONEA, *Ispra*  
A. ERIKSSON, *Stockholm*  
C. FARHAT, *Boulder, CO*  
C.A. FELIPPA, *Boulder, CO*  
C.J. FITZSIMONS, *Baden-Dättwil*  
M. GERADIN, *Liège*  
R. GRUBER, *Manno*  
H.-Å. HÄGGBLAD, *Luleå*  
E.J. HAUG, *Iowa City, IA*

J.C. HEINRICH, *Tucson, AZ*  
U. HEISE, *Aachen*  
J. HELLESLAND, *Oslo*  
C. HOEN, *Oslo*  
M. HOGGE, *Liège*  
S. IDELSOHN, *Santa Fe*  
L. JOHANSSON, *Linköping*  
C. JOHNSON, *Göteborg*  
M. KAWAHARA, *Tokyo*  
S.W. KEY, *Albuquerque, NM*  
A. KLARBRING, *Linköping*  
M. KLEIBER, *Warsaw*  
P. LADEVEZE, *Chachan*  
A. LEGER, *Clamart*  
B.P. LEONARD, *Akron, OH*  
P. LE TALLEC, *Paris*  
W.K. LIU, *Evanston, IL*  
G. MAIER, *Milan*  
H.A. MANG, *Vienna*  
A. NEEDLEMAN, *Providence, RI*  
M.P. NIELSEN, *Lyngby*

A.K. NOOR, *Hampton, VA*  
R. OHAYON, *Paris*  
J. PERIAUX, *Saint Cloud*  
QIAN Ling-xi (L.H. Tsien), *Dalian*  
A.K. RAO, *Hyderabad*  
B.D. REDDY, *Rondebosch*  
J.N. REDDY, *College Station, TX*  
E. RIKS, *Delft*  
G.I.N. ROZVANY, *Essen*  
W. SCHIEHLEN, *Stuttgart*  
M.S. SHEPARD, *Troy, NY*  
E. STEIN, *Hannover*  
P.K. SWEBY, *Reading*  
M. TANAKA, *Nagano*  
T.E. TEZDUYAR, *Houston, TX*  
C.W. TROWBRIDGE, *Kidlington*  
H. VAN DER VORST, *Utrecht*  
J.R. WHITEMAN, *Uxbridge*  
K.J. WILLAM, *Boulder, CO*  
T. ZIMMERMANN, *Lausanne*

Editorial Secretary: Marlies PARSONS

**Advertising information.** Advertising orders and enquiries can be sent to: **Europe and ROW:** Rachel Gresle-Farthing, Elsevier Science Ltd., Advertising Department, The Boulevard, Langford Lane, Kidlington, Oxford OX5 1GB, UK; phone: (+44) (1865) 843565; fax: (+44) (1865) 843976; e-mail: r.gresle-farthing@elsevier.co.uk. **USA and Canada:** Elsevier Science Inc., Mr Tino DeCarlo, 655 Avenue of the Americas, New York, NY 10010-5107, USA; phone: (+1) (212) 633 3815; fax: (+1) (212) 633 3820; e-mail: t.decarlo@elsevier.com. **Japan:** Elsevier Science K.K., Advertising Department, 9-15 Higashi-Azabu 1-chome, Minato-ku, Tokyo 106, Japan; phone: (+81) (3) 5561-5033; fax: (+81) (3) 5561 5047.

© The paper used in this publication meets the requirements of ANSI/NISO Z39.48-1992 (Permanence of Paper).

# Renumbering strategies for unstructured-grid solvers operating on shared-memory, cache-based parallel machines

Rainald Löhner

*GMU/CSI, The George Mason University, Fairfax, VA 22030-4444, USA*

Received 30 July 1996; revised 18 June 1997

## Abstract

Two renumbering strategies for field solvers based on unstructured grids that operate on shared-memory, cache-based parallel machines are described. Special attention is paid to the avoidance of cache-line overwrite, which can lead to drastic performance degradation on this type of machines. Both renumbering techniques avoid cache-misses and cache-line overwrite while allowing pipelining, leading to optimal coding for this type of hardware. © 1998 Elsevier Science S.A. All rights reserved.

## 1. Introduction

There can hardly be any doubt that the 1990s are the decade of parallelism. Even though machines with several powerful vector-processors were installed at many places in the 1980s, the operating system or the system administration hardly allowed for the use of several processors during the same run. Moreover, in many instances the compilers were not mature, leading to meaningless gains in performance. With the advent of massively parallel machines, i.e. machines in excess of 500 nodes, the exploitation of parallelism in solvers has become a major focus of attention. According to Amdahl's Law, the speed-up  $s$  obtained by parallelizing a portion  $\alpha$  of all operations required is given by

$$s = \frac{1}{\alpha \cdot \frac{R_s}{R_p} + (1 - \alpha)}, \quad (1)$$

where  $R_s$ ,  $R_p$  denote the scalar and parallel processing rates (speeds), respectively. Table 1 shows the speed-ups obtained for different percentages of parallelization and different numbers of processors.

Note that even on a traditional shared-memory, multiprocessor vector machine, such as the CRAY T-90 with 16 processors, the maximum achievable speed-up between scalar code and parallel vector code is a staggering  $R_p/R_s = 240$ . What is important to note is that as we migrate to higher numbers of processors, only the

Table 1  
 Speed-ups obtainable (Amdahl's Law)

$R_p/R_s$	50%	90%	99%	99.9%
10	1.81	5.26	9.17	9.91
100	1.98	9.90	50.25	90.99
1000	2.00	9.91	90.99	500.25

embarrassingly parallel codes will survive. Most of the applications ported successfully to parallel machines to date have followed the Single Program Multiple Data (SPMD) paradigm. For grid-based solvers, a spatial subdomain was stored and updated in each processor. For particle solvers, groups of particles were stored and updated in each processor. For obvious reasons, load balancing [1–4] has been a major focus of activity.

Despite the striking successes reported to date, only the simplest of all solvers: explicit timestepping or implicit iterative schemes, perhaps with multigrid added on, have been ported without major changes and/or problems to massively parallel machines with distributed memory. Many code options that are essential for realistic simulations are not easy to parallelize on this type of machine. Among these, we mention local remeshing [5], repeated  $h$ -refinement, such as required for transient problems [6], contact detection and force evaluation [7], some preconditioners [8], applications where particles, flow, and chemistry interact, and applications with rapidly varying load imbalances. Even if 99% of all operations required by these codes can be parallelized, the maximum achievable gain will be restricted to 1:100. If we accept as a fact that for most large-scale codes we may not be able to parallelize more than 99% of all operations, the shared memory paradigm, discarded for a while as non-scalable, will make a comeback. It is far easier to parallelize some of the more complex algorithms, as well as cases with large load imbalance, on a shared memory machine. And it is within present technological reach to achieve a 100 processor, shared memory machine. Such an alternative, i.e. having less expensive RISC chips linked via shared memory, is currently being explored by a number of vendors. One example of such machines is the SGI Power Challenge, which at the time of writing allows up to 18 processors to work in shared memory mode on a problem, with upgrades to 92 processors planned within the next two years. In order to obtain proper performance from such a machine, the codes must be written in such a way as to avoid:

- (a) Cache-misses (in order to perform well on each processor);
- (b) Cache overwrite (in order to perform well in parallel); and
- (c) Memory contention (in order to allow pipelining).

Thus, although in principle a good compromise, shared memory, RISC-based parallel machines actually require a fair degree of knowledge and reprogramming for codes to run optimally.

The present paper describes renumbering techniques for field solvers operating on unstructured grids that have proven useful for machines of this kind. A summary of the remainder of the paper follows. Sections 2 and 3 recall some previously described renumbering techniques to minimize cache-misses and achieve pipelining. Section 4 treats cache overwrite, a new, and previously not accounted-for design requirement for renumbering strategies. In Section 5, implementational issues are considered. Section 6 reports several scalability studies obtained on SGI Power Challenge and SGI Origin systems. Finally, some conclusions are drawn in Section 7.

## 2. Renumbering to avoid cache misses

Consider the following loop over edges that typifies the central loop of many field solvers based on unstructured grids [9–14]. Similar loops are obtained for element- or face-based solvers, and what follows is equally applicable to them. A right-hand side (RHS), or residual, is formed at the edge-level by gathering information from a vector of unknowns. This edge-RHS is then added to a global point-RHS. The operations are shown schematically in Fig. 1, and a typical FORTRAN implementation would be the following:

*Loop 1*

```

do 1600 iedge=1,nedge
  ipoi1=lneod(1,iedge)
  ipoi2=lneod(2,iedge)
  redge=geoed( iedge) * (unkno(ipoi2)-unkno(ipoi1))
  rhspo(ipoi1)=rhspo(ipoi1) + redge
  rhspo(ipoi2)=rhspo(ipoi2) - redge
1600 continue

```

If cache-misses are a concern, then it is clear that the storage locations for the required point information stored

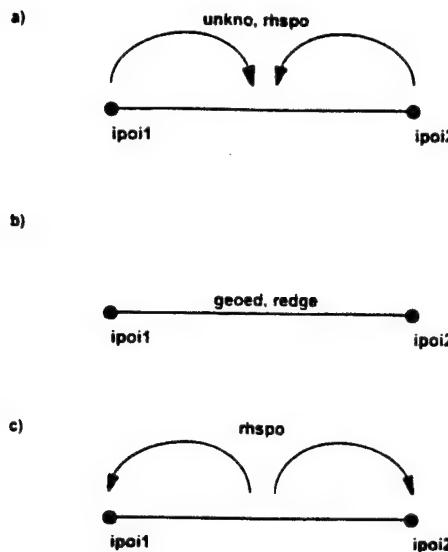


Fig. 1. Information flow for edge-based loop.

in the arrays *unkno* and *rhspo* should be as close as possible in memory when required by an edge. At the same time, as the loop progresses through the edges, the point information should be accessed as uniformly as possible. This may be achieved by first renumbering the points using a bandwidth-minimization technique (e.g. Reverse Cuthill and McKee [15], wavefront [16], recursive bisection), and subsequently renumbering the edges according to the minimum point number on each edge [16]. All of these algorithms are of complexity  $O(N)$  or at most  $O(N \log N)$ , and are well worth the effort.

### 3. Avoidance of memory contention

Pipelining or vectorization offers the possibility of substantial performance gain on any kind of system. While previously restricted to so-called vector machines, such as those manufactured by CRAY, Convex, NEC, Fujitsu or Hitachi, the concept has migrated to current RISC chips, such as the MIPS R8000 and R10000. For the latter chip, so-called software pipelining is invoked by the compiler for certain optimization options. In order to achieve pipelining or vectorization, memory contention issues must be avoided. The enforcement of pipelining or vectorization is then carried out using a compiler directive, as Loop 1, which becomes an inner loop, still offers the possibility of memory contention. In this case, we would have:

#### Loop 2

```

do 1400 ipass=1,npass
nedg0=edpas(ipass) + 1
nedg1=edpas(ipass + 1)
c$dir ivdep                      ! Pipelining directive
do 1600 iedge=nedg0,nedg1
  ipoi1=lnoed(1,iedge)
  ipoi2=lnoed(2,iedge)
  redge=geoed( iedge) * (unkno(ipoi2) - unkno(ipoi1))
  rhspo(ipoi1)=rhspo(ipoi1) + redge
  rhspo(ipoi2)=rhspo(ipoi2) - redge
1600  continue
1400  continue

```

It is clear that in order to avoid memory contention, for each of the groups of edges (1600 loop), none of the

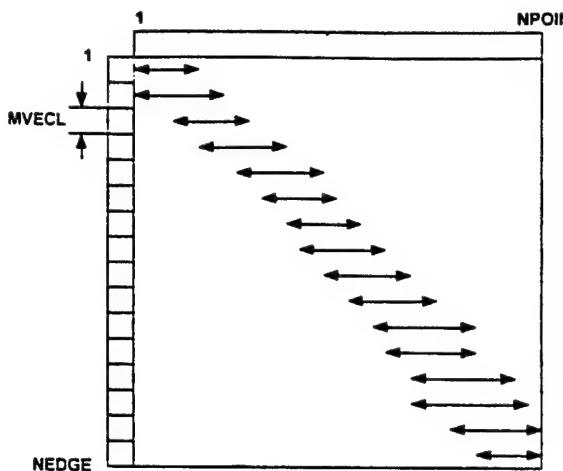


Fig. 2. Point-range covered by each group of edges; 1-Processor machine; renumbering to minimize cache-misses and avoid memory contention.

corresponding points may be accessed more than once. Given that in order to achieve good pipelining performance on current RISC-chips a relatively short vector length of 16 is sufficient, one can simply start from the edge-renumbering obtained in order to minimize cache-misses, and renumber it further into groups of edges that are 16 long and avoid memory contention [16]. As before, this renumbering is of complexity  $O(N)$ . The resulting loop is shown schematically in Fig. 2.

#### 4. Cache line overwrite

The next stage is to port Loop 2 to a parallel, shared memory machine. If the loop is left untouched, the auto-parallelizing compiler will simply split the inner loop across processors. It would then appear that increasing the vector-length to a sufficiently large value would offer a satisfactory solution. However, this is not advisable for the following reasons:

- Every time a parallel do-loop is launched, a start-up time penalty, equivalent to several hundred Flops is incurred. This implies that if scalability to even 16 processors is to be achieved, the vector loop lengths would have to be  $16 * 1000$ . For typical tetrahedral grids we encounter approximately 22 maximum vector-length groups, indicating that we would need at least  $22 * 16 * 1000 = 352\,000$  edges to run efficiently.
- Because the range of points in each group increases at least linearly with vector length, so do cache misses. This implies that even though one may gain parallelism, the individual processor performance would degrade. The end result is a very limited, non-scalable gain in performance.
- Because the points in a split group access a large portion of the edge-array, different processors may be accessing the same cache-line. When a 'dirty cache-line' overwrite occurs, all processors must update this line, leading to a large increase of interprocessor communication, severe performance degradation and non-scalability. Experiments on an 8-processor SGI Power Challenge showed a maximum speed-up of only 1:2.5 when using this option. This limited speed-up was attributed, to a large extent, to cache-line overwrites.

In view of these consequences, additional renumbering strategies have to be implemented. In the following, we discuss two edge-group agglomeration techniques that minimize cache misses, allow for pipelining on each processor, and avoid cache overwrite across processors. Both techniques operate on the premise that the points accessed within each parallel inner edge-loop (1600 loop) do not overlap.

Before going on, we define  $edmin(1:npass)$ ,  $edmax(1:npass)$  to be the minimum and maximum point accessed within each group,  $nproc$  the number of processors, and the point-range of each group  $ipass$  as  $[edmin(ipass), edmax(ipass)]$ .

#### 4.1. Local agglomeration

The first way of achieving pipelining and parallelization is by processing, in parallel, nproc independent vector-groups whose individual point-range does not overlap. The idea is to renumber the edges in such a way that nproc groups are joined together where, for each one of these groups, the point ranges do not overlap (see Fig. 3). As each one of the sub-groups has the same number of edges, the load is balanced across the processors. The actual loop is given by

##### Loop 3

```

do 1400 impass=1,npass
  nedg0=edpas(ipass) + 1
  nedg1=edpas(ipass + 1)
c                                     ! Parallelization directive
c$doacross local(iedge,ipoil,ipois,redge)
c$dir ivdep                         ! Pipelining directive
  do 1600 iedge=nedg0,nedg1
    ipoil=lnoed(1,iedge)
    ipois=lnoed(2,iedge)
    redge=geoed( iedge) * (unkno(ipois) - unkno(ipoil))
    rhspo(ipoil)=rhspo(ipoil) + redge
    rhspo(ipois)=rhspo(ipois) - redge
  1600  continue
  1400  continue

```

Note that the number of edges in each pass, i.e. the difference nedg1 – nedg0 + 1 is now nproc times as large as in the original Loop 2. As one can see, this type of renumbering entails no code modifications, making it very attractive for large production codes. However, a start-up cost is incurred whenever a loop across processors is launched. This would indicate that long vector-lengths should be favoured. However, cache-misses increase with vector-length, so that this strategy only yields a limited speed-up. This leads us to the second renumbering strategy.

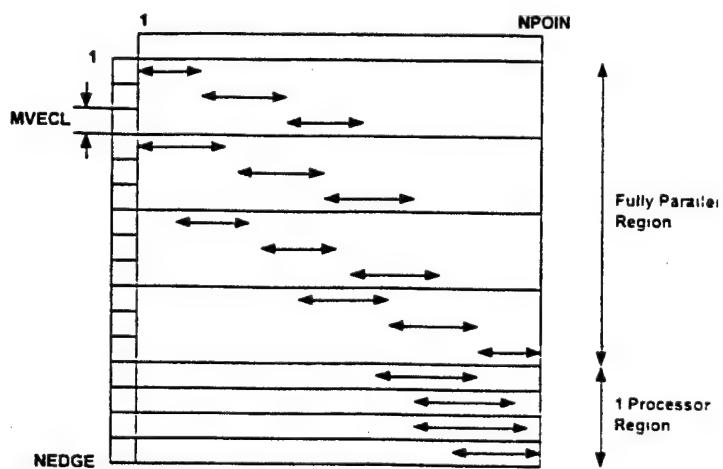


Fig. 3. Point-range covered by each group of edges; 3-Processor machine; renumbering to minimize cache-misses and avoid memory contention.

#### 4.2. Global agglomeration

A second way of achieving pipelining and parallelization is by processing all the individual vector-groups in parallel at a higher level (see Fig. 4). In this way, short vector lengths can be kept and low start-up costs are achieved. As before, the point-range between macro-groups must not overlap. This renumbering of edges is similar to domain-splitting [1,4,17–19], but does not require an explicit message passing or actual identification of domains. Rather, all the operations are kept at the (algebraic) array level. The number of sub-groups, as well as the total number of edges to be processed in each macro-group, is not the same. However, the imbalance is small, and does not affect performance significantly. The actual loop is given by

##### Loop 4

```

do 1000 imacg=1,npasg,nproc
  imac0=           imacg
  imac1=min(npasg,imac0 + nproc - 1)
c                                     ! Parallelization directive
c$doacross local(ipasg,ipass,npas0,
c$&           npas1,iedge,nedg0,nedg1,
c$&           ipo1,ipo12,redge)
  do 1200 ipasg=imac0,imac1
    npas0=edpag(ipasg) + 1
    npas1=edpag(ipasg + 1)
    do 1400 ipass=npas0,npas1
      nedg0=edpas(ipass) + 1
      nedg1=edpas(ipass + 1)
c$dir ivdep                         ! Pipelining directive
  do 1600 iedge=nedg0,nedg1
    ipo1=lnoed(1,iedge)
    ipo12=lnoed(2,iedge)
    redge=geoed( iedge) * (unkno(ipo12) - unkno(ipo1))
    rhspo(ipo1)=rhspo(ipo1) + redge
    rhspo(ipo12)=rhspo(ipo12) - redge
1600      continue
1400      continue
1200      continue
1000      continue

```

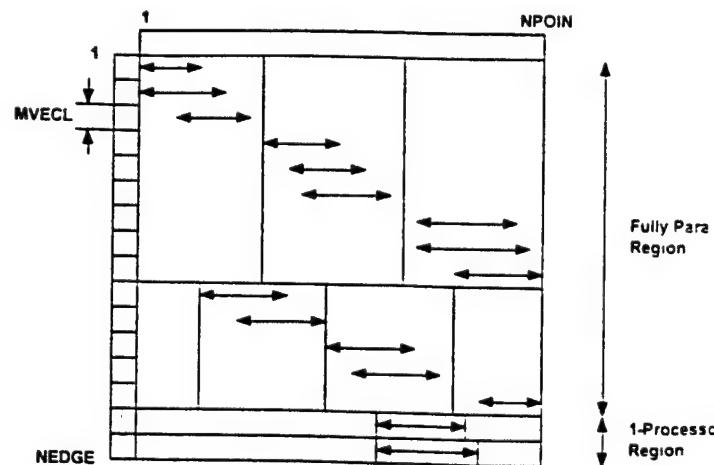


Fig. 4. Point-range covered by each group of edges; 3-processor machine; renumbering to minimize cache-misses, Avoid memory contention and minimize start-up costs.

As one can see, this type of renumbering entails two outer loops, implying that a certain amount of code rewrite is required. On the other hand, the original code can easily be retrieved by setting `edpag(1)=0, edpag(2)=npas, npasg=1` for conventional uniprocessor machines.

If the length of the cache-line is known, one may relax the restriction of non-overlapping point ranges to non-overlapping cache-line ranges. This allows for more flexibility, and leads to almost perfect load balance for all cases tested to date.

A simple edge-renumbering scheme that we have found effective is the following multipass algorithm:

**S.1. Pass 1:** Agglomerate in groups of edges with point-range `npoin/nproc`, setting the maximum number of groups in each macro-group `nagg1` to the number of groups obtained for the range `1:npoin/nproc`;

**S.2. Passes 2ff:**

- For the remaining groups: determine the point-range;
- Estimate the range of the first next macro-group from the point range and the number of processors;
- Agglomerate the groups in this range to obtain the first next macro-group, and determine the number of groups for each macro-group in this pass `nagg1`;
- March though the remaining groups of edges, attempting to obtain macro-groups of length `nagg1`.

Although not optimal, this simple strategy yields balanced macro-groups, as can be seen from the examples below. Obviously, other renumbering or load balancing algorithms are possible, as evidenced by a large body of literature (see e.g. [1,4,17–19]).

## 5. Implementational issues

For large-scale codes, having to re-write and test several hundred subroutines can be an onerous burden. To make matters worse, present compilers force the user to declare explicitly the local and shared variables. This is easily done for simple loops such as the one described above, but can become involved for the complex loops with many scalar temporaries that characterize advanced CFD schemes written for optimal cache reuse. We have found that in some cases, compilers may refuse to parallelize code that has all variables declared properly. A technique that has always worked, and that reduces the amount of variables to be declared, is to write sub-subroutines. For Loop 4, this translates into:

### Master Loop 4

```

do 1000 imacg=1,npasg, proc
  imac0=           imacg
  imac1=min(npasg,imac0 + nproc - 1)
c                                     ! Parallelization directive
c$doacross local(ipasg)
  do 1200 ipasg=imac0,imac1
    call loop2p(ipasg)
1200  continue
1000 continue

```

Loop 2p becomes a subroutine of the form:

```

subroutine loop2p(ipasg)
npas0=edpag(ipasg) + 1
npasi=edpag(ipasg + 1)
do 1400 ipass=npas0,npas1
  nedg0=edpas(ipass) + 1
  nedg1=edpas(ipass + 1)
c$dir ivdep
  do 1600 iedge=nedg0,nedg1           ! Pipelining directive

```

```

ipoi1=lnoed(1,iedge)
ipoi2=lnoed(2,iedge)
redge=geoed(  iedge)*(unkno(ipoi2)-unkno(ipoi1))
rhspe(ipoi1)=rhspe(ipoi1)+redge
rhspe(ipoi2)=rhspe(ipoi2)-redge
1600  continue
1400  continue

```

## 6. Example timings

The renumbering strategies described were coded into FEFLO97, an adaptive, edge-based finite element code for the solution of compressible and incompressible flows [20]. The compressible solver incorporates, among other options, van Leer's flux-vector and Roe's flux-difference splitting techniques. The incompressible solver is based on a projection technique, implying that the bulk of the CPU time is spent in a Laplacian loop of precisely the type discussed above. The first two cases were run on an SGI Power Challenge with 6 R8000 processors, 4 Mbytes of cache and 512 Mbytes of memory, whereas the third case was run on an SGI Origin 2000 system with 8 R10000 processors, 4 Mbytes of cache and 4 Gbytes of memory.

### 6.1. F-117

The surface mesh, as well as the (unconverged) solution after 50 timesteps are shown in Fig. 5(a,b). The mesh had approximately 280 Ktetra, 52 Kpts, 8.6 Kboundary points and 340 Kedges. After renumbering, the

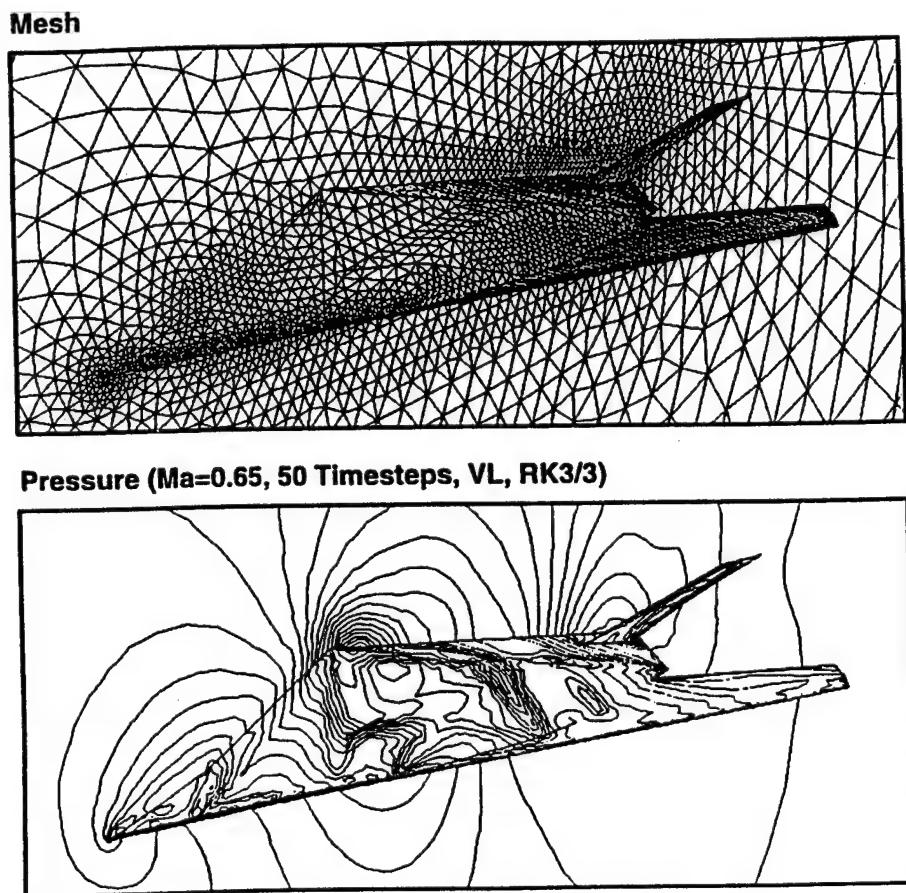


Fig. 5. F-117 (transonic). (a) Surface mesh; (b) pressure ( $Ma_\infty = 0.65$ ).

Table 2

ipasg	npas0	npas1	loop1	ipmin	ipmax
(a) F-117 problem on 2 processors					
1	1	8705	139280	1	23588
2	8706	17410	139280	23588	46686
3	17411	18936	24416	19980	27264
4	18937	20462	24416	45465	50160
5	20463	20760	4768	49486	50832
6	20761	21058	4767	50875	51874
7	21059	21263	3280	50319	51320
(b) F-117 problem on 4 processors					
1	1	3791	60656	1	10704
2	3792	7582	60656	10709	23020
3	7583	11373	60656	23029	35623
4	11374	15164	60656	35631	46685
5	15165	16054	14240	8665	13101
6	16055	16944	14240	19470	25301
7	16945	17834	14240	32241	37526
8	17835	18724	14240	45458	48692
9	18725	19231	8112	19430	26447
10	19232	19738	8112	34507	48782
11	19739	20245	8112	48784	50727
12	20246	20590	5519	50729	51874
13	20591	20724	2144	22706	48803
14	20725	20858	2144	50168	51041
17	20859	21263	6480	47928	51226
(c) F-117 problem on 6 processors					
1	1	2160	34560	1	6282
2	2161	4320	34560	6413	13918
3	4321	6480	34560	14021	22627
4	6481	8640	34560	22668	31484
5	8641	10800	34560	31487	39596
6	10801	12960	34560	39751	46585
7	12961	13555	9520	4858	7897
8	13556	14150	9520	12696	17070
9	14151	14745	9520	20533	25686
10	14746	15340	9520	28369	33391
11	15341	15935	9520	36851	40814
12	15936	16530	9520	45325	47920
13	16531	17161	10096	6247	15424
14	17162	17792	10096	19030	24188
15	17793	18423	10096	27860	34282
16	18424	19054	10096	38363	48191
17	19055	19685	10096	48192	50496
18	19686	20121	6975	50497	51874
19	20122	20273	2432	12674	25973
20	20274	20425	2432	30955	34624
21	20426	20577	2432	47184	48525
22	20578	20729	2432	49873	50832
25	20730	20857	2048	22256	26268
26	20858	20985	2048	31320	48657
27	20986	21052	1072	50319	51007
31	21053	21180	2048	22573	48853
37	21181	21263	1328	47985	49065
(d) F-117: Actual vs. optimal edge-allocation					
nproc	actual %	optimal %	loss %		
2	50.482	50.00	1.0		
4	26.934	25.00	7.7		
6	18.234	16.67	9.4		
(e) Timings for F-117 problem					
nproc	time (s)	CPU (s)	Speedup		
1	919	897.0	1.00		
2	485	942.5	1.89		
4	281	1087.5	3.27		
6	220	1235.1	4.17		

maximum and average bandwidths were 3797 and 2510. The vector-loop length was set to 16, which was found to be sufficient for good performance on the SGI Power Challenge. The grouping of edges according to the number of processors is given in Table 2(a–c). The corresponding percentage of edges processed by the processor with the maximum number of edges, as well as the theoretical loss of performance due to imbalance (ratio of actual work carried out by this processor vs. the minimum possible work) is shown in Table 2(d). Table 2(e) summarizes the clock time and total CPU time, as well as the speed-ups obtained for a run of 50 timesteps, including i/o, renumbering, etc. As one can see, the performance degrades with the number of processors. This is to be expected, as the increasing number of passes results in higher relative loop costs, and portions of the code (i/o, renumbering, indirect data structures, some residual sums, etc.) are still running in uni-processor mode. Given that the machine used only had 6 processors, timings obtained for the 6 processor case may be higher than expected.

### 6.2. Sphere

The surface mesh, as well as the solution after 50 timesteps are shown in Fig. 6(a,b). The mesh had approximately 332 Ktetra, 61 Kpts, 8.8 Kboundary points and 402 Kedges. After renumbering, the maximum and average bandwidths were 1993 and 1411. The vector-loop length was again set to 16. The grouping of edges according to the number of processors is given in Table 3(a–c). The corresponding percentage of edges processed by the processor with the highest number of edges, as well as the theoretical loss of performance due to imbalance is shown in Table 3(d). Table 3(e) summarizes the speed-ups obtained for a run of 50 timesteps, including i/o, renumbering, etc. Note that the speed-up is superlinear up to four processors. A convincing explanation of this phenomenon is still elusive, but we speculate on reduced cache-misses for the multiprocessor runs. As before, the machine used had a total of 6 processors, so that the timings for the 6 processor case have to be qualified. The bulk of the work for this incompressible flow case is performed in a Laplacian-like inner loop over edges, showing that the data structures discussed perform well.

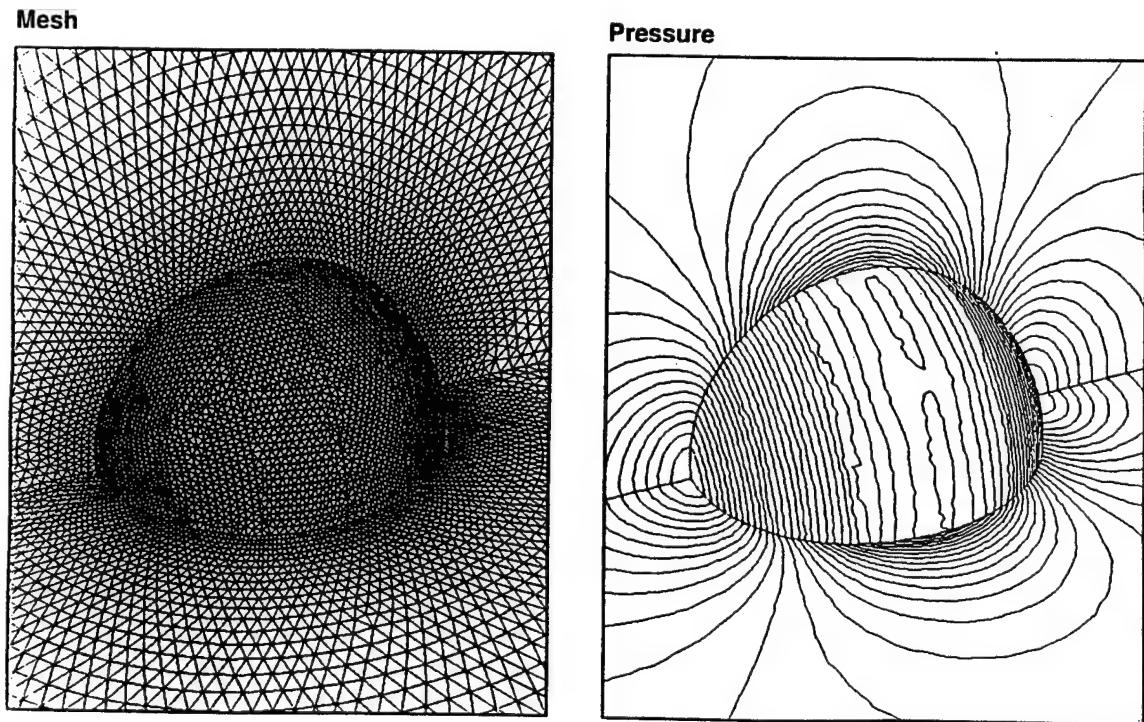


Fig. 6. Sphere (incompressible). (a) Surface mesh; (b) pressure.

Table 3

ipasg	npas0	npas1	loop1	ipmin	ipmax
(a) Sphere problem on 2 processors					
1	1	10753	172048	1	27373
2	10754	21506	172048	27377	54935
3	21507	22282	12416	25525	29268
4	22283	23058	12416	53374	56594
5	23059	23937	14064	55413	58529
6	23938	24816	14062	58530	61039
7	24817	35118	4832	57736	59207
(b) Sphere problem on 4 processors					
1	1	5023	80368	1	13171
2	5024	10046	80368	13172	26969
3	10047	15069	80368	26970	40962
4	15070	20092	80368	40964	54771
5	20093	21097	16080	11892	28113
6	21098	22102	16080	39029	55218
7	22103	23107	16080	55219	58651
8	23108	23940	13326	58654	61039
9	23941	24175	3760	26232	28688
10	24176	24410	3760	53705	55676
11	24411	24645	3760	57869	59179
13	24646	24773	2048	26788	55787
14	24774	24836	1008	58484	59308
17	24837	25118	4512	54486	56437
(c) Sphere problem on 6 processors					
1	1	3349	53584	1	8960
2	3350	6698	53584	8962	18476
3	6699	10047	53584	18478	28352
4	10048	13396	53584	28354	38356
5	13397	16745	53584	38357	48329
6	16746	18797	32832	48330	54935
7	18798	19639	13472	7924	19449
8	19640	20481	13472	26448	38468
9	20482	21323	13472	46448	55065
10	21324	22165	13472	55066	58148
11	22166	23007	13472	58149	60703
12	23008	23094	1390	60705	61039
13	23095	23363	4304	17870	38498
14	23364	23632	4304	53557	55636
15	23633	23901	4304	57295	58737
16	23902	24001	1600	60427	60914
19	24002	24149	2368	36539	38829
20	24150	24297	2368	54246	55940
21	24298	24354	912	57991	58859
25	24355	24482	2048	36906	39125
26	24483	24610	2048	54651	56231
31	24611	24738	2048	37201	39439
32	24739	24765	432	54986	56291
37	24766	25118	5648	37506	40314
(d) Sphere: Actual vs. optimal edge-allocation					
nproc	actual %	optimal %	loss %		
2	50.601	50.00	1.2		
4	26.567	25.00	6.2		
6	20.770	16.67	24.6		
(e) Timings for sphere problem					
nproc	time (s)	CPU (s)	Speedup		
1	2259	2204.0	1.00		
2	1050	2043.2	2.15		
4	532	2065.0	4.25		
6	430	2493.3	5.25		

Table 4

ipasg	min (loop1)	max (loop1)	
(a) Inlet problem on 2 processors			
1–2	237776	237776	
3–4	71344	71344	
5–6	21392	21392	
7–8	6416	6416	
9–10	1770	1936	
11–12	112	1024	
13–14	0	576	
(b) Inlet problem on 4 processors			
1–4	118880	118880	
5–8	35664	35664	
9–12	10704	10704	
13–16	3216	3216	
17–20	266	1024	
21–24	800	1024	
25–28	0	256	
(c) Inlet problem on 6 processors			
1–6	79248	79248	
7–12	23776	23776	
13–18	7136	7136	
19–24	0	2144	
25–30	1024	1024	
31–36	0	1024	
37–42	0	336	
(d) Inlet problem on 8 processors			
1–8	59440	59440	
9–16	17824	17824	
17–24	5360	5360	
25–32	650	1600	
33–40	0	1024	
41–48	0	1024	
49–56	0	288	
(e) Sphere: Actual vs. optimal edge-allocation			
nproc	actual %	optimal %	loss %
2	50.122	50.00	0.24
4	25.140	25.00	0.56
6	16.884	16.67	1.01
8	12.743	12.50	1.94
(f) Speedups for inlet problem			
nproc	Speedup (Shared)	Speedup (PVM)	
2	1.81	1.83	
4	3.18	3.50	
6	4.31	5.10	
8	5.28	—	

### 6.3. Supersonic inlet

The surface definition, as well as the solution after 800 timesteps are shown in Fig. 7(a,b). The mesh had approximately 540 Ktetra, 106 Kpts, 30 Kboundary points and 680 Kedges. After renumbering, the maximum and average bandwidths were 1057 and 468. The vector-loop length was again set to 16. Instead of showing detailed tables as before, only the minimum and maximum number of edges processed for each pass over the

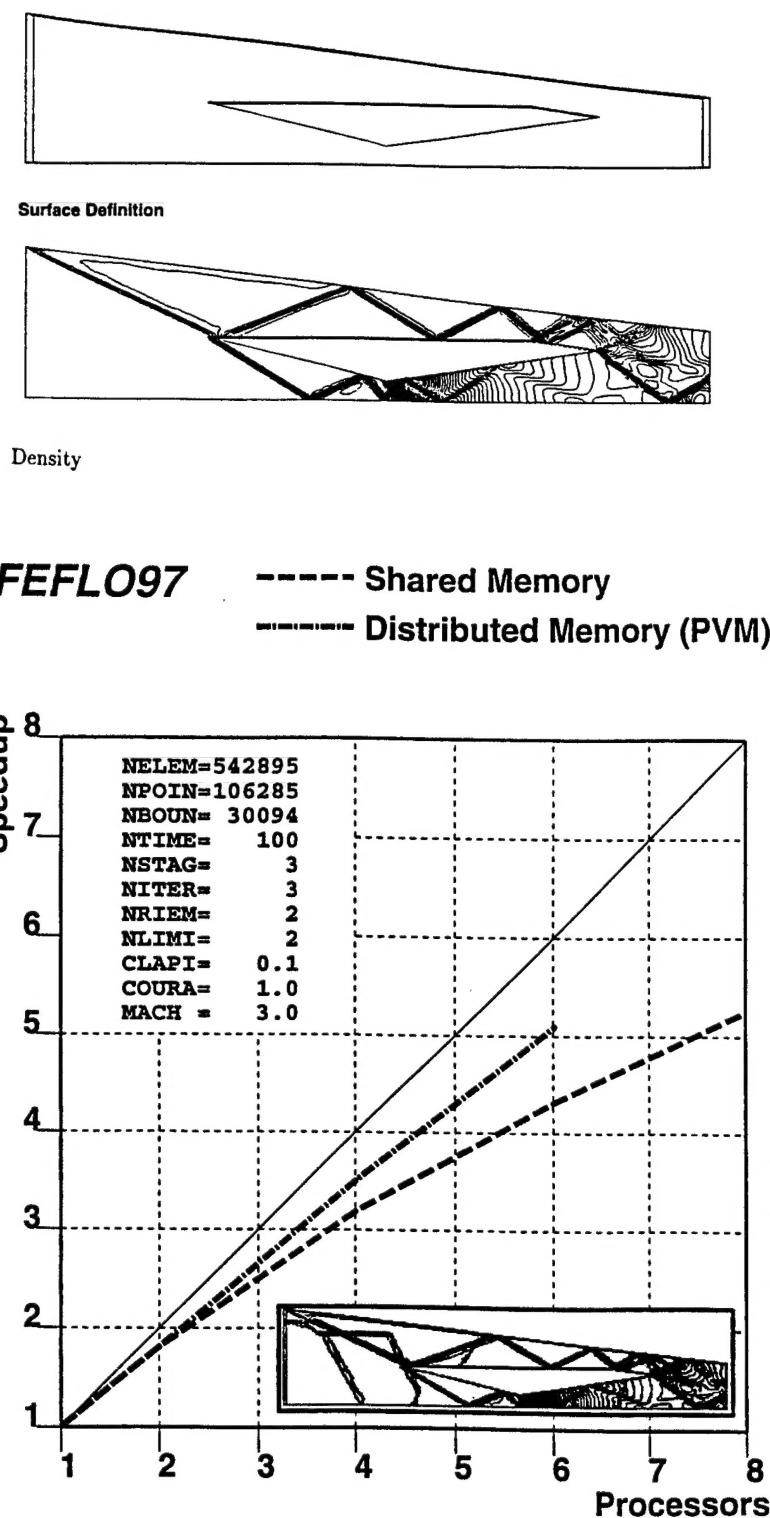


Fig. 7. Inlet problem (supersonic). (a) Surface definition; (b) density; (c) speedups.

processors is given in Table 4(a–d). The corresponding percentage of edges processed by the first processor, as well as the theoretical loss of performance due to imbalance is shown in Table 4(e). As compared to the previous two examples, a near optimal load balance is achieved. Two reasons may be given for this improvement. First, this example was run with a newer version of the renumbering techniques. Second, the constraint of monotonicity in the range of points covered by each macro-group of edges was relaxed to a non-overlap at the cache-line level, which for the SGI Power Challenge is about 15 reals. Table 4(f) and Fig. 7(c) summarize the speed-ups obtained for a run of 100 timesteps, including i/o, renumbering, etc. for an SGI Origin 2000 machine. Although this machine is not a true shared-memory machine, it exhibits very fast inter-processor transfer rates, making it possible to achieve reasonable speedups in shared memory mode. The same run was repeated using domain decomposition and message passing under PVM. Observe that although the PVM-run achieves better speedup, the shared memory run is still competitive.

## 7. Conclusions

Two renumbering strategies for field solvers based on unstructured grids that operate on shared-memory, cache-based parallel machines have been described. Special attention was given to the avoidance of cache-line overwrite, a hitherto not considered design requirement, which, if not taken into account, can lead to drastic performance degradation on this type of machine. Both renumbering techniques avoid cache-misses and cache-line overwrite while allowing pipelining, leading to optimal coding. While the first technique requires no code rewrite of the field solver, its scalability is expected to degrade for a large number of processors. The second technique requires a moderate rewrite of traditional field solvers, but offers the potential of near-linear scalability for a large number of processors and problem sizes. Numerical experiments indicate that with these renumbering techniques, the number of passes over the processors is always below the theoretical minimum number of passes one would require for maximum-length loops, which for tetrahedral meshes is 7. This implies that these techniques are also applicable to static memory machines like the CRAY-T90, reducing loop start-up costs and improving performance as compared to straightforward inner loop autotasking. As with any other technique, improvements and variations are possible. The techniques described will, however, work on any shared-memory, cache-based machine, and are in this sense general.

## Acknowledgments

This work was partially supported by AFOSR, with Dr. Leonidas Sakell as the technical monitor. The author would also like to acknowledge the many fruitful discussions with Drs. Jan Clinkemaille (ESI Group, Paris, France), Jeffrey D. McDonald (SGI, Mountain View, CA), Jack Perry (SGI, Boston, MA), as well as Wayne Odachowsky (SGI, Bethesda, MD), who was instrumental in coordinating the collaboration that led to the techniques discussed.

## References

- [1] D. Williams, Performance of dynamic load balancing algorithms for unstructured grid calculations, CalTech Rep. C3P913 (1990).
- [2] H. Simon, Partitioning of unstructured problems for parallel processing, NASA Ames Tech. Rep. RNR-91-008 (1991).
- [3] P. Mehrota, J. Saltz and R. Voigt, eds., Unstructured Scientific Computation on Scalable Multiprocessors (MIT Press, 1992).
- [4] A. Vidwans, Y. Kallinderis and V. Venkatakrishnan, A parallel load balancing algorithm for 3-D adaptive unstructured grids, AIAA-93-3313-CP (1993).
- [5] R. Löhner, Three-dimensional fluid–structure interaction using a finite element solver and adaptive remeshing, Computer Syst. Engrg. (2–4) (1990) 257–272.
- [6] R. Löhner and J.D. Baum, Adaptive H-refinement on 3-D unstructured grids for transient problems, Int. J. Numer. Methods Fluids 14 (1992) 1407–1419.
- [7] E. Haug, H. Charlier, J. Clinckemaillie, E. DiPasquale, O. Fort, D. Lasry, G. Milcent, X. Ni, A.K. Pickett and R. Hoffmann, Recent trends and developments of crashworthiness simulation methodologies and their integration into the industrial vehicle design cycle, Proc. Third European Cars/Trucks Simulation Symposium (ASIMUTH), Oct. 28–30, 1991.

- [8] R. Ramamurti and R. Löhner, Simulation of flow past complex geometries using a parallel implicit incompressible flow solver, Proc. 11th AIAA CFD Conf., Orlando, FL (July 1993) 1049, 1050.
- [9] T. Barth, A 3-D Upwind Euler solver for unstructured meshes, AIAA-91-1548-CP, 1991.
- [10] D. Mavriplis, Three-dimensional unstructured multigrid for the Euler equations, AIAA-91-1549-CP (1991).
- [11] A. Jameson, The AIRPLANE Code, Private communication, January 1992.
- [12] J. Peraire, J. Peiro and K. Morgan, A three-dimensional finite element multigrid solver for the Euler equations, AIAA-92-0449 (1992).
- [13] H. Luo, J.D. Baum, R. Löhner and J. Cabello, Adaptive edge-based finite element schemes for the Euler and Navier–Stokes equations, AIAA-93-0336 (1993).
- [14] N.P. Weatherill, O. Hassan and D.L. Marcum, Calculation of steady compressible flowfields with the finite element method, AIAA-93-0341 (1993).
- [15] E. Cuthill and J. McKee, Reducing the bandwidth of sparse symmetric matrices, Proc. ACM Nat. Conf., New York (1969) 157–172.
- [16] R. Löhner, Some useful renumbering strategies for unstructured grids, Int. J. Numer. Methods Engrg. 36 (1993) 3259–3270.
- [17] N. Satofuka, J. Periaux and A. Ecer, eds., Parallel Computational Fluid Dynamics (North-Holland, 1995).
- [18] V. Venkatakrishnan, H.D. Simon and T.J. Barth, A MIMD implementation of a parallel Euler solver for unstructured grids, NASA Ames Tech. Rep. RNR-91-024 (1991).
- [19] R. Löhner and R. Ramamurti, A load balancing algorithm for unstructured grids, Comput. Fluid Dyn. 5 (1995) 39–58.
- [20] R. Löhner, FEFLO97 Theoretical Manual; GMU-CSI-CFD Lab. Report, 1996.

## INFORMATION FOR CONTRIBUTORS

Manuscripts should be sent in triplicate to one of the Editors. All manuscripts will be refereed. Manuscripts should preferably be in English. They should be typewritten, double-spaced, first copies (or clear Xerox copies thereof) with a wide margin. Abstracts, footnotes and lists of references should also be double-spaced. All pages should be numbered (also those containing references, tables and figure captions). Upon acceptance of an article, author(s) will be asked to transfer copyright of the article to the publisher. This transfer will ensure the widest possible dissemination of information.

### Abstracts

The text of a paper should be preceded by a summary in English. This should be short, but should mention all essential points of the paper.

### Figures and tables

The drawings for the figures must be originals, drawn in black India ink in large size and carefully lettered, or printed on a high-quality laser printer. The lettering as well as the details should have proportionate dimensions, so as not to become illegible or unclear after the usual reduction by the printers; in general, the figures should be designed for a reduction factor of two or three. Mathematical symbols should be entered in italics, where appropriate. Each figure should have a number and a caption; the captions should be collected on a separate sheet. The appropriate place of a figure should be indicated in the margin. Tables should be typed on separate sheets. Each table should have a number and a title. The appropriate places for the insertion of tables should be indicated in the margin. Colour illustrations can be included and will be printed in colour at no charge if, in the opinion of the Editors, the colour is essential. If this is not the case, the figures will be printed in black and white unless the author is prepared to pay the extra costs arising from colour reproduction.

### Formulae

Displayed formulae should be numbered and typed or clearly written by hand. Symbols should be identified in the margin, where they occur for the first time.

### References

In the text, reference to other parts of the paper should be made by section (or equation) number, but not by page number. References should be listed on a separate sheet in the order in which they appear in the text.

COMPLETE INSTRUCTIONS TO AUTHORS are published in every last issue of a volume, and copies can also be obtained from the Editors and the Publisher, Elsevier Science B.V., P.O. Box 1991, 1000 BZ Amsterdam, The Netherlands.

### Instructions for LaTeX manuscripts

The LaTeX files of papers that have been accepted for publication may be sent to the Publisher by e-mail or on a diskette (3.5" or 5.25" MS-DOS). If the file is suitable, proofs will be produced without rekeying the text. The article should be encoded in Elsevier-LaTeX, standard LaTeX, or AMS-LaTeX (in document style "article"). The Elsevier-LaTeX package, together with instructions on how to prepare a file, is available from the Publisher. This package can also be obtained through the Elsevier WWW home page (<http://www.elsevier.nl/>), or using anonymous FTP from the Comprehensive TeX Archive Network (CTAN). The host-names are: `ftp.dante.de`, `ftp.tex.ac.uk`, `ftp.shsu.edu`; the CTAN directories are: `/pub/tex/macos/latex209/contrib/elsevier`, `/pub/archive/macos/latex209/contrib/elsevier`, `/tex-archive/macos/latex209/contrib/elsevier`, respectively. *No changes from the accepted version are permissible, without the explicit approval of the Editor. The Publisher reserves the right to decide whether to use the author's file or not.* If the file is sent by e-mail, the name of the journal should be mentioned in the "subject field" of the message to identify the paper. Authors should include an ASCII table (available from the Publisher) in their files to enable the detection of transmission errors.

### Publication information:

*Computer Methods in Applied Mechanics and Engineering* (ISSN 0045-7825). For 1998 volumes 151–163 are scheduled for publication. Subscription prices are available upon request from the Publisher. Subscriptions are accepted on a prepaid basis only and are entered on a calendar year basis. Issues are sent by surface mail except to the following countries where Air delivery via SAL mail is ensured: Argentina, Australia, Brazil, Canada, Hong Kong, India, Israel, Japan, Malaysia, Mexico, New Zealand, Pakistan, PR China, Singapore, South Africa, South Korea, Taiwan, Thailand, USA. For all other countries airmail rates are available upon request. Claims for missing issues should be made within six months of our publication (mailing) date.

### Orders, claims, and product enquiries:

please contact the Customer Support Department at the Regional Sales Office nearest you:  
**New York:** Elsevier Science, PO Box 945, New York, NY 10159-0945, USA; phone: (+1) (212) 633 3730 [toll free number for North American customers: 1-888-4ES-INFO (437-4636)]; fax: (+1) (212) 633 3680; e-mail: [usinfo-f@elsevier.com](mailto:usinfo-f@elsevier.com)

**Amsterdam:** Elsevier Science, PO Box 211, 1000 AE Amsterdam, The Netherlands; phone: (+31) 20 4853757; fax: (+31) 20 4853432; e-mail: [nlinfo-f@elsevier.nl](mailto:nlinfo-f@elsevier.nl)

**Tokyo:** Elsevier Science K.K., 9-15 Higashi-Azabu 1-chome, Minato-ku, Tokyo 106, Japan; phone: (+81) (3) 5561 5033; fax: (+81) (3) 5561 5047; e-mail: [info@elsevier.co.jp](mailto:info@elsevier.co.jp)

**Singapore:** Elsevier Science, No. 1 Temasek Avenue, #17-01 Millenia Tower, Singapore 039192; phone: (+65) 434 3727; fax: (+65) 337 2230; e-mail: [asiainfo@elsevier.com.sg](mailto:asiainfo@elsevier.com.sg)

**Rio de Janeiro:** Elsevier Science, Rua Sete de Setembro 111/16 Andar, 20050-002 Centro, Rio de Janeiro – RJ, Brazil; phone: (+55) (21) 509 5340; fax: (+55) (21) 507 1991; e-mail: [elsevier@campus.com.br](mailto:elsevier@campus.com.br) [Note (Latin America): for orders, claims and help desk information, please contact the Regional Sales Office in New York as listed above]